NASA Technical Memorandum 87635

THE LATDYN USER'S MANUAL

(NASA-TM-87635) THE LATDYN USER'S MANUAL N86-21953
(NASA) 212 p HC A10/MF A01 CSCL 20K

Unclas
G3/39 05846

JERROLD M. HOUSNER, PAUL E. MCGOWAN,
A. LOUIS ABRAHAMSON, AND MICHAEL G. POWELL

JANUARY 1986

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

THE LATDYN USER'S MANUAL

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

ACKNOWLEDGEMENTS

The authors would like to acknowledge Bradford Bingel of Computer Sciences Corporation and Maria Mitchum of the Systems Engineering Division at the NASA Langley Research Center for their dedicated assistance in creating the pre- and post-processing portions of the LATDYN program.

SECTION 1 - INTRODUCTION

The purpose of this manual is to present the capabilities of the LATDYN (Large Angle Transient DYNamics) computer program and the instructions necessary for its efficient use. Detailed user input and output specifications are provided and discussed, and example runstreams, including selected output are given. The theoretical basis for the computer program may be found in the attached reference document at the end of the manual.

The manual is structured so that it may be used for both tutorial and reference purposes. For tutorial purposes, a new user is advised to read Sections 1 through 4 and then proceed directly to Section 7, to examine and duplicate on their computer the sample problems of that Section. The sample problems are arranged such that in the early problems the more commonly used features of the program are demonstrated and more specialized commands are shown in the later problems. In this way, the user can rapidly learn the commands required for solving most basic problems and then proceed to more complex ones. As a reference manual, a summary of all input commands and their function is provided in Table 4.1-1.

The LATDYN program consists of the following three components:

(1) a pre-processor to verify and interpret input data
(2) a set of computational routines
(3) a post-processor to plot and analyze results

The entire program may be run in one execution or the components may be executed separately. The structure of these three components is more fully described in Appendix A.

To date, LATDYN has been configured for CDC/NOS and the DEC VAX/VMS computers. Operating instructions for executing LATDYN on these computers are provided in Appendices B and C.

All LATDYN coding is done in ANSII-77 FORTRAN. This provides a large measure of portability between computers. Nevertheless, small differences may still exist between versions for different computers. Detailed instructions regarding the interfaces with particular computer operating systems and file structures are contained in Appendices B and C of this manual.

SECTION 2 - CAPABILITIES

The LATDYN program is a tool for analyzing the controlled or uncontrolled dynamic transient behavior of interconnected deformable multi-body systems which can undergo large angular motions of each body relative to other bodies. The program can accommodate large structural deformation as well as large rigid body rotations and is applicable but not limited to the following areas:

(1) large flexible space structures such as lattice or truss type structures which are deployed through unfolding of the lattice deformable beam members from a packaged state

(2) slewing of large space structure components
(3) mechanisms which can be modeled as a collection of springs, masses, viscous dampers and rigid or deformable beam members
(4) robotic manipulations of beam members

The current program is limited to two-dimensional problems, but in many cases, three dimensional problems can be exactly or approximately reduced to two dimensions.

The program is finite element based and utilizes convected coordinates of each finite element to affect the large angular motions involved in the analysis. Because of its finite element nature, the program permits general geometry and users having familiarity with other more general finite element programs should find LATDYN easy to use. The use of finite elements eliminates the user concern with open or closed loop topologies common to other multi-body analysis procedures and therefore this terminology will not be used in the remainder of this document. Those users already familiar with finite element theory may comprehend the analytical manipulations behind the user instructions, but knowledge of finite element theory is not a prerequiste to successful program execution.

One of the complexities encountered in the solution of multi-body transient analysis problems are changes in the structural stiffness and mass properties which occur during the anaylsis. Many of these changes occur due to contact problems, such as encountered in docking and berthing of structural systems, and lock-up of deployable joints during structural deployment. To enable the user to direct such structural changes LATDYN uses a unique flexible control language. It allows the user external access to internal portions of the programs coding, thereby enhancing the program's capabilities.

The beginning user is not required to learn the full capablities of the control language in order to use the program. Rather, it is envisioned that the user will learn the control language progressively.

SECTION 3 - CLASSIFICATIONS OF INPUT DATA

As an aid in discussion, the input is divided into nineteen different classifications as follows:

1. Title *
2. Shorthand Definition
3. Analysis Control *
4. Material Properties *
5. Grid Points *
6. Beam Members *
7. Scaling
8. Springs
9. Viscous Dampers
10. Lumped Masses and Additional Inertial Coupling
11. Preloads
12. Applied Loads
13. Reactions
14. Constraints
15. Initialization
16. Logical Conditions
17. Variable Assignments
18. Lock-Up of Joints
19. Output Control *

Classifications marked by an asterisk (*) indicate that at least some input data within this classification is always required. Each of these classifcations will be discussed in detail in the next section.

No specific order is required for input data, however, a suggested order is employed in the sample problems of Section 7.

SECTION 4 - GENERAL FORMAT OF INPUT DATA

Section 4.1 - Command Names

Input is supplied to the program in the form of commands. Each command may consist of many lines of data with each line being up to 80 columns in length. Each command has an associated alpha-numeric name which the user generally provides in the first line of the command. The user selects from the list of admissible command names in Table 4.1-1. The function of each of these commands is described later. In addition Table 4.1-1 provides a brief format description of each command for quick reference.

Commands which are placed in square brackets, [ ], are not available to the user in this release. Commands which are placed in braces, { }, are post-processor commands. This is explained at greater length on page 5.19-8

The command name commences in the first column of the first line of a command and is followed by a delimeter which may be either a comma or a colon. Following the delimeter, is a sequence of alpha-numeric data values placed in a specific order and separated by commas.

In general the first line of a command takes the form,

```
command name:sequenced data
          or
command name,sequenced data
```

If a name is absent from the first line of a command, it is assumed that the command name is a repeat of the command name which preceeded it. In such a case a delimeter is not required and only the sequenced data is given, such as,

```
command name:sequenced data
sequenced data
```

Blank spaces between data entries are generally ignored, (except that imbedded blanks in an alpha-numeric name or a numerical data value are not permitted), so that it is also permissible to write,

```
command name:sequenced data
             sequenced data
```

which has the same effect as the preceeding example and is useful for visual inspection of data input.

A list of admissable command names is given in Table 4.1-1. The list is broken down into the various classifications of Section 3. Required commands are flagged. Also Table 4.1-2 provides an alphabetical listing of commands with the associated data format. Section 5 provides full documentation and examples for each command.

Table 4.1-1 Command Name Listing by Classification

Table 4.1-1 Command Name Listing by Classification (Continued)

Table 4.1-2 Command Names and Data Format

Command Name	Data Format	Page No.
ADMASS:	grid point number and direction for two degrees-of-freedom which have an additional mass coupling, followed by value of mass coupling ? condition label	5.10-1
APPLOAD:	identification integer,angular orientation, scale factor, time values at which load comes on and off, grid points at which load is to be applied	5.12-1
CLj: or Cj	defines j th condition label using a FORTRAN expression	5.15-1
DAMPER:	grid point numbers at ends of damper, integer specifying orientation,damper constant? condition label	5.9-1
DEFINE:	@n=constant (where n is an integer)	5.2-1
DIS:	grid point number, direction, initial displacement value	5.14-1
FREQ:	integer indicating how often frequencies are calculated,integer specifying how many frequencies are desired, integer specifying how many mode shapes are desired ? condition label	5.3-8
FREQWINDOW:	two frequency values specifying frequency window for PSD command	5.18-20
GRAVITY:	acceleration due to gravity at sea level, X and Y coordinates of earth's center	5.12-8
GRDPRT:	string of grid points at which motions are printed	5.18-7
GRID:	starting grid point number of string, number of grid points in string, x and y global coordinates of first and second grid points in string.(required command)	5.5-1

Unless otherwise noted, the use of a command name is optional.

Table 4.1-2 Command Names and Data Format (Continued)

Command Name	Data Format	Page No.
INCMASS:	time step increment for updating mass matrix, angular rotation for updating mass matrix ? condition label	5.3-1
INIT:	scaling value for initial displacements, scaling value for initial velocities	5.14-1
KEEP:	Qn values to be saved, integer number of previous time steps at which designated Qn values are to be saved	5.16-8
INTGTYP:	integer specifying time integration algorithm	5.3-3
LOAD:	identification integer, load value	5.12-4
LOADSIN:	identification integer, sinusoidal frequency, phase shift and amplitude	5.12-6
LOADTIM:	identification integer, relative or absolute time,pairs of time,load values for piecewise linear definition of load	5.12-8
LOCKUP:	a series of coefficients describing the locking joint model ? condition label	5.18-1
MATPROP:	material name or number, E,A,I,density ? condition label	5.4-1
MDFC:	number of degress-of-freedom linearly constrained together, grid point numbers, directions and multipliers for each degree-of-freedom constrained,right-hand-side of linear constraint relationship ? condition label	5.13-3

Unless otherwise noted, the use of a command name is optional.

Table 4.1-2 Command Names and Data Format (Continued)

Command Name	Data Format	Page No.
MEMBER:	grid point number of ends A and B of finite element, number of finite elements in string, material name or number	5.6-1
MEMPRT:	string of member numbers at which internal forces are printed	5.18-8
PIN:	two grid point numbers which have a connecting pin between them allowing full rotational freedom	5.13-6
PLOTAX:	member numbers in which axial loads are plotted	5.18-16
PLOTC:	list of condition labels whose TRUE-FALSE values are plotted	5.18-16
PLOTFEXT:	M or A for magnitude or orientation angle of plotted external forces, load set identification integers	5.18-15
PLOTINC:	time step for saving data for plotting ? condition label	5.18-11
PLOTLIMIT:	lower and upper time limits for plotting	5.18-11
PLOTMOMA:	member numbers in which moments at end A are plotted	5.18-16
PLOTMOMB:	member numbers in which moments at end B are plotted	5.18-16
PLOTQ:	list of user defined Q-variables whose values are plotted	5.18-16
PLOTQVQ	list of user defined Q-variables which are plotted aganist the first Q-variable of the list	5.18-17

Unless otherwise noted, the use of a command name is optional.

Table 4.1-2 Command Names and Data Format (Continued)

Command Name	Data Format	Page No.
PLOTSHA:	member numbers in which shear forces at end A are plotted	5.18-16
PLOTSHB:	member numbers in which shear forces at end B are plotted	5.18-16
PORDER:	number of finite elements, number of grid points (required command)	5.3-1
PRELOAD:	member number, preload value	5.11-1
PRINT:	time step increment for printing grid point motions, coordinate system for motions,time step increment for printing internal forces, coordinate system for forces, time step increments for printing mass and stiffness matrices ? condition label	5.18-4
PSD:	list of parameters to creat power spectral density plots	5.18-19
PSL:	list of parameters to create power spectral level calculations	5.18-19
RESET:	Qn=FORTRAN expression (n is an integer)	5.16-4
RESTART:	integer indicating restart	5.3-2
ROTLOCK:	two grid point numbers that rotationally lock? condition label	5.17-5

Unless otherwise noted, the use of a command name is optional.

Table 4.1-2 Command Names and Data Format (Continued)

Command Name	Data Format	Page No.
SCALE:	scale for x direction, scale for y direction, rotation	5.7-1
SDFC:	grid point number, direction ? condition label	5.13-1
SET:	Qn=FORTRAN expression (n is an integer)	5.16-2
SNAPSHOT:	list of parameters for plotting a snapshot of structure	5.18-18
SPRING:	grid point numbers at ends of spring, integer specifying spring orientation, spring constant,initial spring stretch ? condition label	5.8-1
STIFF:	integer specifying how often tangent stiffness is to be calculated, Y if incremental stiffness is to be included or N if not ? condition label	5.3-7
STOP:	? condition label	5.3-10
SUBTITLE:	data case subtitle (up to 71 characters)	5.2-1
TIMEWINDOW:	two time values specifying plot time limits	5.18-20
TIMSPAN:	start time, stop time	5.3-6
TIMSTEP:	time step.size ? condition label	5.3-6
TITLE:	data case title (up to 74 characters)	5.1-1
VEL:	grid point number, direction, initial velocity value	5.14-1

Section 4.2 - Continuation Lines

Commands of more than one line in length may be constructed through the use of continuation lines. A continuation line does not commence with the command name, but continues with the data input as though only one line were being used. The line preceeding a continuation line must terminate with an & symbol. The limit on the number of continuation lines that can be used is ten, with the exception that TITLE and SUBTITLE commands do not permit any continuation lines.

In general, a command with continuation lines takes the form,

```
command name: sequenced data &
                sequenced data
```

Note that an input line without a command name is either a continuation line if the last entry of the preceeding line is an & symbol or is the first line of a command with the same name as the preceeding command if the last entry of the preceeding line is not an & symbol.

Section 4.3 - User Supplied Comments

Users are encouraged to make liberal use of comments to annotate their input data. A "$" sign at any location on a line of data indicates that whatever follows on that line is a comment and will be ignored by the program. Commented commands take the general form,

```
command name:sequenced data $comment
```

Continuation lines solely for comments require a "$" in the first column of the continuation line.

The user may wish to separate data types with blank lines by placing a "$" sign in the first column on the line separating the data types as,

```
command name:sequenced data $comment
$
command name:sequenced data $ comment
```

Similarly, the user may wish to tabulate the data associated with a command for ease of visual inspection as,

```
$                heading A      heading B      heading C
command name:     data 1A        data 1B        data 1C
                  data 2A        data 2B        data 2C
                  data 3A        data 3B        data 3C
```

NOTE 4.3-1: The user is cautioned to avoid placing a continuation & symbol after a $ symbol, since it will be interpreted as part of a comment.

Section 4.4 - Conditional Commands

Many of the admissible commands can be made conditional. That is, the user can specify conditions under which the command will be active or inactive. This is accomplished through the use of a condition label which is placed at the end of the command following the last data entry of the command. The condition label is separated from the last data entry by a question mark, "?" such as,

command name:sequenced data ? condition label

A condition label identifies a user supplied TRUE or FALSE logic statement which makes the data preceeding it conditional; that is, the data is used only when the logic statement is TRUE. The logic statement is written in FORTRAN V and can make use of variables calculated during program execution such as displacement, velocity, acceleration, member strain, etc. A complete list of variables available to the user is provided in Table 4.4-1.

The employment of condition labels gives the user the flexibility of specifying such things as contact conditions during impact phenomena, lock-up conditions for joints in deployable structures, control of parameters used in data output and a host of other functions. Section 5.15 provides further information on the use of condition labels.

Table 4.4-1 Admissible Variables For Condition Labels and User Defined Variables

Variable Name	Explanation
XLOC(n)	for the current location of grid point n parallel to the global x axis
YLOC(n)	for the current location of grid point n parallel to the global y axis
ROT(m)	for the current rotated position of member m measured positive in a direction from the positive global x to the positive global y axis
X(p,n)	for component of motion quantity (displacement, if p=D; velocity, if p=V; acceleration, if p=A) at grid point n parallel to the global x axis
Y(p,n)	for component of motion quantity (displacement, if p=D; velocity, if p=V; acceleration, if p=A) at grid point n parallel to the global y axis
XP(p,n,l)	for component of motion quantity (displacement, if p=D; velocity, if p=V; acceleration, if p=A) at grid point n parallel to a local x'axis (parallel to a designated member, l)
YP(p,n,l)	for component of motion quantity (displacement, if p=D; velocity, if p=V; acceleration, if p=A) at grid point n parallel to a local y'axis (transverse to designated member, l)

where,

p is a parameter indicating the type of motion desired as,

D, for displacement

V, for velocity

A, for acceleration

l designates the member number associated with local axes

m is a member number

n is a grid point number

Table 4.4-1 Admissible Variables For Condition Labels and User Defined Variables (continued)

Variable Name	Explanation
PHI(p,n)	for rotation at grid point n in the positive angular direction; from positive global x axis to positive global y axis
MAG(p,n)	for magnitude of motion quantity (displacement, if p=D; velocity, if p=V; acceleration, if p=A) at grid point n
ANGLE(p,n)	for angular orientation of motion quantity (displacement, if p=D; velocity, if p=V; acceleration, if p=A) at grid point n; positive angles are measured from positive global x axis towards positive global y axis
PI	for π
PIH	for $\pi/2$
PIQ	for $\pi/4$
PI2	for 2π

where,

p is a parameter indicating the type of motion desired as,

D, for displacement

V, for velocity

A, for acceleration

n is a grid point number

Table 4.4-1 Admissible Variables For Condition Labels and User Defined Variables (continued)

Variable Name	Explanation
AX(m)	for axial force in designated member m (positive for tension)
SHA(m)	for shear force at end A of beam member m
SHB(m)	for shear force at end B of beam member m
BM(m,sbar)	for bending moment in m^{th} beam member at a dimensionless distance sbar (0 < sbar < 1) from end A of the member
FEXTX(j)	for component of j^{th} external force parallel to the global x axis
FEXTY(j)	for component of j^{th} external force parallel to global y axis
FEXTM(j)	for the magnitude of the j^{th} external force
FEXTA(j)	for angular orientation of the j^{th} external force; positive angles are measured from the positive global x to positive global y axis

where,

m is a member number

j refers to the applied load specified by the j^{th} APPLOAD command

sbar designates location at which bending moment is desired as the distance from end A of the beam ratioed to the beam length

Table 4.4-1 Admissible Variables For Condition Labels and User Defined Variables (continued)

Variable Name	Explanation
REACTX(n)	for reaction force at grid point n in the global x-direction
REACTY(n)	for reaction force at grid point n in the global x-direction
REACTT(n)	for reaction torque at grid point n in the global x-direction
REACTM(n)	for the magnitude of the reaction force at grid point n
REACTA(n)	for the angular orientation of the angular force at grid point n
REACTC(n,l)	for the component of the reaction force at grid point n parallel to the local coordinate system of member l
STRAIN(m)	for axial strain in member m
T	for time
NTSTEP	for time step number
OMEGA(i)	for frequency in radians per unit time of designated mode i, where the absolute value of i designates the i th lowest mode if i is negative and the i th highest mode if i is positive
CLj or Cj	for the j^{th} logical condition; provides logical TRUE or FALSE for condition label Cj
QH(Qk,r)	for the value of user defined variable Qk at a time r time steps prior to the current value of time

where,

m is a member number

i is a mode number

r is number of time steps prior to current time

Section 4.5 - Multiple Occurrences of a Command Name

The user may supply multiples of most commands. For example, there may be any number of GRID commands in the input data. However, for the following command names, only one command can be active at a time: TIMSTEP, INCMASS, STIFF, FREQ, MATPROP, PRINT and PLOTINC. To determine which command of a multiple set is active, the program uses a condition-order priority as explained in NOTE 5.15-1 of Section 5.15. When multiples of these commands are employed, the first of these commands which appears in the data set cannot have an attached condition label, while the subsequent ones must each have an attached condition label. The preprocessor will search for violations of this rule and if any violations are found, execution will terminate.

Section 4.6 - Units for Input Data

Any consistent set of units may be used with the exception that angular measurements must be in radians and frequencies in Hertz.

SECTION 5 - INPUT SPECIFICATIONS

Section 5.1 - Title

Titles for data case identification are specified by using the commands, TITLE and SUBTITLE. Character strings following either of these commands may contain any ASCII standard characters in any order, but may not exceed one line in length (80 characters) including command name. The designated title will appear at the beginning of the printed and plotted output. Use of the TITLE command is required, but use of the SUBTITLE command is optional. Data takes the form,

TITLE, free field descriptor
SUBTITLE, free field descriptor

Example 5.1-1

```
TITLE, DEPLOYMENT OF 10 BAY TRUSS
SUBTITLE, UNIFORM PROPERTIES
```

Section 5.2 - Shorthand Definition

The DEFINE and LIST commands are optional and may be used to define constants or lists of integers which often appear in the input data. Other commands can use these values in a shorthand fashion.

DEFINE

Shorthand definitions begin with the name DEFINE. This is followed by a colon or comma an "@" sign, an integer, an equal sign and the specified constant value. Defined constants may then be used in other commands by using the "@" sign followed by the appropriate integer in place of a numerical value. Data takes the form,

DEFINE,@n=constant

where,

n is an integer

Example 5.2-1: Defining constants

```
DEFINE,@1=3.14159265359$ DEFINE PI
DEFINE, @99= 1.4142136$ DEFINE SQUARE ROOT OF 2
```

A condition label cannot be used in conjunction with a DEFINE command.

NOTE 5.2-1: The user may define up to 99 constants.

NOTE 5.2-2: The @n may appear in place of any numerical value in nearly all commands. Forms of @n requiring a sign change are used as "-@n".

LIST

The list command is an optional command used to define a sequence of integers which can be used in a shorthand fashion in certain commands. At present, only the APPLOAD, GRDPRT, MEMPRT and LIST commands are able to use the defined lists created by a LIST command. Data takes the form,

LIST,n,m1,m2,m3,...

A LIST command may also use the symbols T and B. The T implies "through" and is used to define a string of quantities, and the B implies "by" and is used to designate the spacing between subsequent items in the list.

Example 5.2-2: Defining a list of integers

LIST, 1,1,2,4T10
In this example a list of integers 1,2,4,5,6,7,8,9,10 is defined

Example 5.2-3: Defining a list of equally spaced integers

LIST,20T30B
In this example, a list of integers is defined with spacing of two, namely, 20,22,24,26,28 and 30.

NOTE 5.2-3: Floating point values used in a LIST command are truncated to integer values.

NOTE 5.2-4: A LIST may not specify more then fifty integers.

Section 5.3 - Analysis Control

Several commands are used to specify parameters controlling program execution. Some of these commands are required; others are optional. The following commands are employed:

PORDER

This is a required command which defines the maximum structural model size as the number of finite elements in the model and the number of grid points in the model. Data takes the form,

PORDER, maximum number of elements, maximum number of grid points

Example 5.3-1: Specifying the maximum problem size .

```
PORDER,50,120$ A MAXIMUM OF 50 FINITE ELEMENTS AND
$120 GRID POINTS ARE ALLOCATED IN THIS EXAMPLE
```

In this example, computer storage requirements for the problem are allocated to allow for a maximum of 50 finite beam elements and 120 grid points.

A condition label cannot be used with this command.

NOTE 5.3-1: The values specified by the user must not be exceeded in the geometry of the problem. Since the computer will be instructed to set aside memory sufficient to handle the problem dimensions, it is efficient, but not necessary, to set the maximum problem dimensions equal to those actually existing in the structural model.

RESTART

This is an optional command which indicates if a restart case is being executed or if a restart tape is to be generated. Data takes the form,

RESTART,IN - indicates that a restart case is being executed

RESTART,OUT - indicates that a restart tape is to be generated

RESTART,BOTH - indicates that a restart case is being executed and that a restart tape is to be generated.

A condition label cannot be used in conjunction with a RESTART command. If a RESTART command does not appear in the input data, it is assumed that neither is a restart case being executed nor a subsequent restart envisioned.

INTGTYP

This is an optional command which indicates the time integration algorithm to be employed in the analysis . Data takes the form,

INTGTYP,1 - indicates the use of central differences

INTGTYP,2 - indicates the use of the Newmark Beta method in an explicit fashion with beta equal to unity.

INTGTYP,3 - indicates Adams-Bashforth predictor in conjunction with Adams-Moulton corrector and automatic variable step size

INTGTYP,4 - indicates use of fifth order Runge Kutta with automatic variable step size

INTGTYP,5,β - indicates the use of the Newmark Beta method in an implicit fashion. If the value of beta is left blank, a default value of 1/3 is used.

If the INTGTYP command is omitted from the data set, the default is the Newmark Beta method in an explicit fashion with beta equal to unity, INTGTYP=2. Condition labels cannot be used with this command.

NOTE 5.3-2: INTGTYP values of 2 through 5 are not available in the current version of LATDYN.

INCMASS

This is an optional command used to designate updating of the system mass matrix. Data takes the form,

INCMASS,n,angle ? Cj

where,

n is a positive integer specifying that the mass matrix is to be periodically updated with a periodicity of n time steps; if zero or blank periodic updating of the mass matrix will not occur

angle is the maximum magnitude of angular motion, in radians, that any rigid member, (see MEMBER command in section 5.4), will be permitted to rotate without there having occured an updating of the mass matrix; if "angle" is zero or left blank, a default value of .0872 radians (5 degrees) is assumed

Cj is the j th optional condition label defined with a Cj command; see Section 5.15 for further details.

Example 5.3-2: Specifying mass matrix updates

```
INCMASS,100,0.2$ MASS MATRIX UPDATED EVERY 100 TIME
$STEPS
```

In this example, the mass matrix is updated every 100 time steps; however, it will be updated sooner if any rigid member of the structure rotates plus or minus 0.2 radians since the previous mass matrix update.

NOTE 5.3-3: Updating of the mass matrix will occur every n time steps, where the length of the time step is that value which is currently being used by the program. In addition, updating of the mass matrix will occur when the increment of angular motion of a rigid member, since the most recent update, (independent of what caused the update), reaches the specified "angle" value.

NOTE 5.3-4: Multiple INCMASS commands may be used, but they have a condition-order priority as discussed in NOTE 5.15-1 of Section 5.15. Thus, the first appearance of an INCMASS command in the data set cannot have an attached condition label. See also Section 4.5.

NOTE 5.3-5: If the INCMASS command is omitted from the data set, the mass matrix will not be updated on a periodic basis. However, events which can lead to large changes in the mass matrix such as impact or lock-up of joints are sensed by the program and result in automatic updating of the mass matrix. The printed output will indicate this occurrence. (If rigid elements are not employed in the modeling, periodic mass matrix updating is usually not necessary and this command may be omitted if there is no other reason to update the mass matrix. For further guidelines on updating the mass matrix, see Section 6.2)

NOTE 5.3-6: Mass matrix updating affects the velocities of the structure. Structural velocities will be automatically altered so as to conserve the momentum of the system. See section 6.2 for further discussion.

TIMSPAN

This is a required command which supplies the starting and terminating times of the transient response. Data takes the form,

```
TIMSPAN,t1,t2
```

where,

t1 is the starting time for the transient response
t2 is the final time for the transient response

Example 5.3-3: Setting the time interval to be analyzed

```
TIMSPAN,0.0,200.$ 200 SECONDS OF RESPONSE STARTING
$AT 0.
```

A condition label cannot be used with the TIMSPAN command.

TIMSTEP

This is a required command which supplies the time integration step and takes the form,

```
TIMSTEP,delt ? Cj
```

where,

delt is a time increment
Cj is the j th optional condition label defined with a Cj command; see section 5.15 for further details

Example 5.3-4: Setting the time step

```
TIMSTEP,1.E-4$ TIME INCREMENT IS 0.0001 TIME UNITS
```

NOTE 5.3-5: Multiple TIMSTEP commands may be used, but they have a condition-order priority as discussed in NOTE 5.15-1 of Section 5.15. Thus, the first appearance of a TIMSTEP command in the data set cannot have an attached condition label. See also Section 4.5.

STIFF

This is an optional command indicating the calculation of the system tangent stiffness matrix. The program does not use the stiffness matrix to predict the time history of response, but it may be calculated for a variety of other reasons depending on user preference. For example, the user may wish to determine the system frequencies or may wish to use the stiffness to examine structural or control stability. Data takes the form,

STIFF,n,p ? Cj
where,

n is an integer specifying that the tangent stiffness is to be up-dated every n time steps; if n=0 then the stiffness matrix is up-dated automatically when it is required for calculation of system frequencies

p is a parameter specifying that the incremental stiffness, arising from the axial load carried by the member, is to be included or excluded from the stiffness matrix. If p is Y (for yes) the incremental stiffness is included in the stiffness matrix calculation; if p is N (for no), the incremental stiffness is excluded from the stiffness matrix calculation.

Cj is the j th optional condition label defined with a Cj command; see section 5.15 for further details

Example 5.3-5: Creating the system tangent stiffness matrix

```
STIFF,1000,Y$ STIFFNESS CALCULATED EVERY 1000 STEPS
```

In this example, the tangent stiffness including the effect of incremental stiffness is calculated every 1000 time steps.

NOTE 5.3-6: Multiple STIFF commands may be used, but they have a condition-order priority as discussed in NOTE 5.15-1 of Section 5.15. Thus, the first appearance of a STIFF command in the data set cannot have an attached condition label. See also Section 4.5.

FREQ

This is an optional command for calculating instantaneous frequencies and associated mode shapes of the system. (The method of bisection is employed to extract frequencies.) These calculations require the mass and tangent stiffness matrices of the system. The mass matrix is automatically calculated, but since the transient algorithm used in the program does not use the tangent stiffness matrix, it is not calculated unless the user requests its calculation through the STIFF command. Thus when using the FREQ command, the STIFF command must also be used. The program recalculates the matrices whenever frequencies are required. Data takes the form,

FREQ,k,m,n ? Cj

where,

k is an integer specifying frequency and mode shape calculation and automatic output every k time steps

|m| is an integer specifying the number of lowest frequencies or highest frequencies desired; if m is negative the lowest |m| frequencies are calculated and if m is positive, the highest m frequencies are calculated; of m=0, no frequencies are calculated, but if m is blank, then all frequencies are calculated

n is an integer < |m| specifying the number of mode shapes desired; n mode shapes associated with the lowest frequencies, if m is negative, the highest frequencies if m is prositive and all mode shapes if n is blank

Cj is the j th optional condition label defined with a Cj command; see section 5.15 for further details

Example 5.3-6: Calculation of system frequencies

```
FREQ,1500,-10,10$ 10 LOWEST FREQUENCIES CALCULATED
```

In this example, the 10 lowest frequencies and mode shapes are calculated every 1500 time steps. In performing this calculation, the program will use the most recently up-dated mass and stiffness matrices.

NOTE 5.3-8: Multiple FREQ commands may be used, but they have a condition-order priority as discussed in NOTE 5.15-1 of Section 5.15. Thus, the first appearance of an FREQ command in the data set cannot have an attached condition label. See also Section 4.5.

STOP

This is an optional command which if used always appears with a condition label. It instructs the termination of the problem when the condition label is true and thus may be used to terminate job execution before maximum time in TIMSPAN command is reached. Data takes the form,

STOP ? Cj

where,

Cj is the j th condition label defined with a Cj command; see section 5.15 for further details

NOTE 5.3-10: Section 5.15 on condition labels provides additional information concerning this command and appropriate examples.

Section 5.4 - Beam Material and Structural Properties

At present only spring, damper and flexible beam members are available. Spring and damper properties are described under their own subheading. Beam material and structural properties are specified using the command MATPROP. This command is required if beam members are used. Data takes the form,

MATPROP:matname,mtype,E,A,I,rho ? Cj

where,

matname is a unique property identification integer or alpha-numeric name

mtype is the type of beam member with four beam types being permitted

- mtype=0, flexible member
- mtype=1, extensionally rigid, but flexible in bending
- mtype=2, flexurally rigid, extensionally flexible
- mtype=3, fully rigid

(Presently only mtype =0 and 3 are operational.)

If mtype entry is blank, the default value is zero.

E is Young's modulus

A is the beam cross sectional area

I is the beam area moment of inertia

rho is the mass density per unit volume

Cj is the j th optional condition label defined with a Cj command; see section 5.15 for further details

NOTE 5.4-1: Up to 99 different beam properties can be specified. Any consistent set of units may be used in the MATPROP command.

Example 5.4-1: Aluminum and graphite-epoxy properties

```
MATPROP,ALUMINUM,0,10.5E6,.1,.01,2.588E-4$ DEFINES
$ ALUMINUM MATERIAL PROPERTIES
GRAPHITE/EPOXY,0,40.E6,.2,.01,2.E-4$ DEFINES
$GRAPHITE-EPOXY MATERIAL PROPERTIES
```

In this example, two beam material properties are defined. The first material corresponds to aluminum beam members and the presence of zero in the second numerical entry indicates that beam members which refer to this MATPROP command are fully flexible.

The third line of this example does not commence with a command name. Since the numerical data of the first line does not terminate with a & sign, the program recognizes that the third line constitutes a new command which is of the same type as the most previous command, namely, a MATPROP command. (Had the previous line of data ended with a & sign, the program would have assumed the third line of the example to be a continuation line of the first.) The second MATPROP command would be referred to by beam members which are fully flexible and made of graphite-epoxy

Example 5.4-2: Material Properties of Rigid Members

```
MATPROP,3,3,,,,2.E-4$ MATERIAL 3 REFERS TO FULLY
$ RIGID MEMBERS
4,3,,,,5.E-4$ MATERIAL 4 REFERS TO FULLY RIGID
$ MEMBERS
```

In this example, two material properties are defined. Beam members which refer to these property commands are fully rigid.

NOTE 5.4-2: There may be only eight MATPROP commands with the same identification number and with attached condition labels and there must be at least one MATPROP command without a condition label for each material identification number appearing in the input data.

NOTE 5.4-3: Multiple INCMASS commands with the same identification number or name may be used, but they have a condition-order priority as discussed in NOTE 5.15-1 of Section 5.15. Thus, the first appearance in the data set of an INCMASS command, with a particular identification number or name, cannot have an attached condition label. See also Section 4.5.

NOTE 5.4-4: When rigid members are employed, MTYPE greater than zero, it is advisable to periodically update the mass matrix. (See INCMASS command in Section 5.2 and user guidelines in Section 6.2 for further details.)

NOTE 5.4-5: When the mass property of a member is changed during exectuion, the program will automatically update the mass matrix and adjust the velocitities of the structure to conserve system momentum. Energy, however, may not be conserved. See section 6.2 for further discussion.

Section 5.5 - Grid Point Locations and Stationary Global Coordinate System

The stationary global cartesian coordinate system (X,Y) is depicted in figure 5.5-1. Locations of grid points in this system are specified with the GRID command. A single grid point or a continuous string of grid points may be specified with one GRID command.

The GRID command automatically generates grid points in an equally spaced pattern between two user specified grid points. Grid point numbers are automatically assigned sequentially with a user supplied increment. Data takes the form,

GRID,nst,nn,x1,y1,x2,y2,inc

where,

- nst is the starting grid point number
- nn is the number of grid points defined by this string
- x1 and y1 are the x and y coordinates of the first grid point in the string
- x2 and y2 are the x and y coordinates of the last grid point in the string
- inc is the increment in the numbering of the grid points

To define only one grid point, nn is set to 1, inc is set to zero or blank, and x2 and y2 are left blank. If values for x2 and y2 are given, they are ignored by the program when nn is 1.

Example 5.5-1: Define Strings of Grid Point Coordinates

```
GRID, 10,11,100., 100., 1000., 1000.,10$ FIRST
$STRING
110, 3,100., 100., 1000.,  100.,10$ SECOND STRING
```

In this example, two strings of grid points are defined. The first GRID command defines the first string with grid point numbers 10, 20, 30,...,100 and coordinates (100.,100.), (200.,200.), 300.,300.),...,(1000.,1000.), respectively. The second line is also a GRID command and defines the second string with grid point numbers 110, 120, 130, and coordinates (100.,100.), (500.,100.) (1000.,100.).

NOTE 5.5-1: An unlimited number of GRID commands may be used, but the total number of grid points defined by all GRID commands cannot exceed the maximum value designated with the PORDER command. At least one GRID command must be used. If more than one GRID command defines the same grid point, the coordinates of the last command are used.

A condition label cannot be used in conjunction with a GRID command.

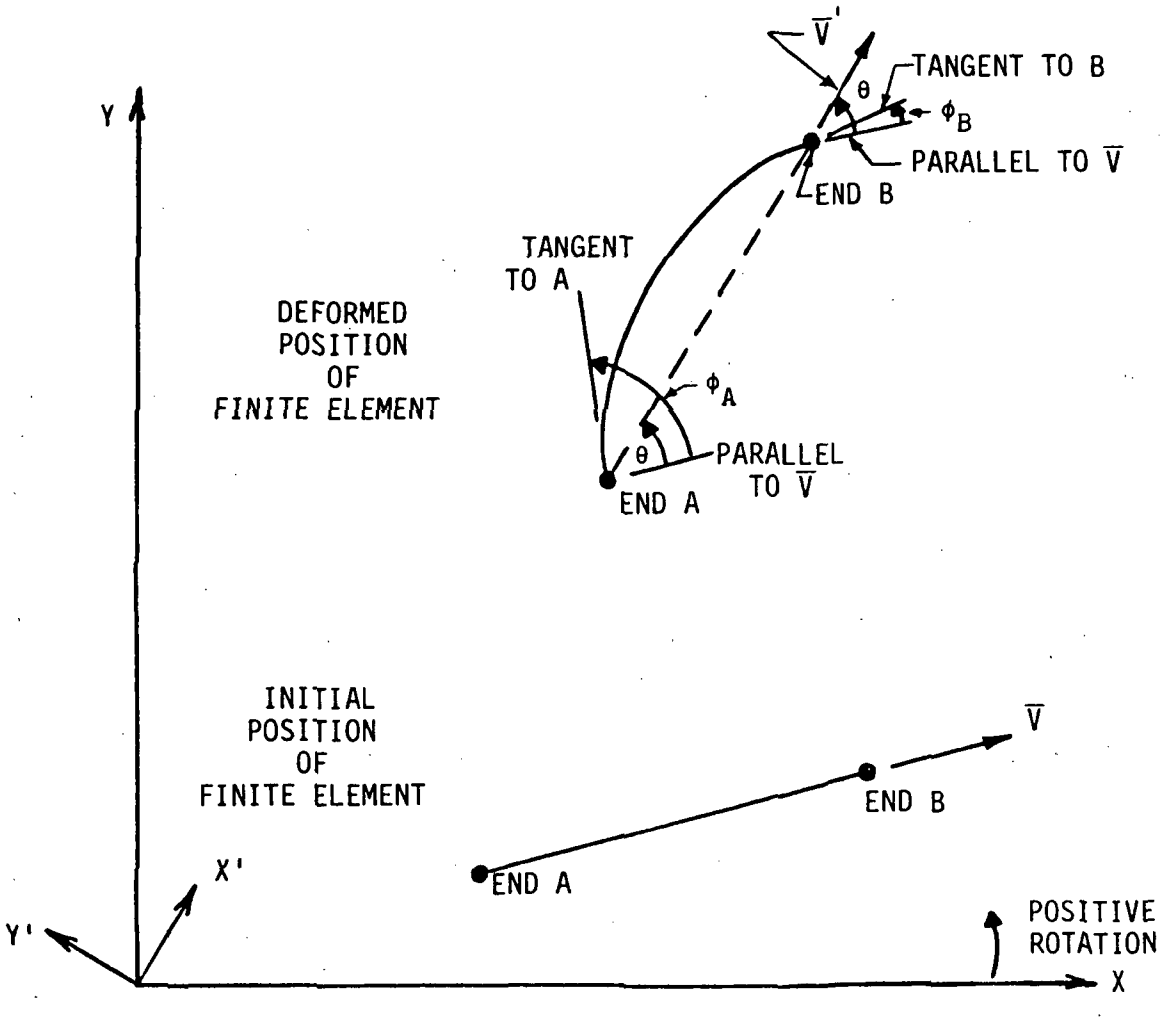


Figure 5.5-1 Global (X,Y) and Local (X',Y') Coordinate Systems

Section 5.6 - Beam Member and Local Coordinate Definition

5.6.1 - Beam Members

Beam members are defined using the MEMBER command. A single member or a continuous string of members may be defined with a single MEMBER command. The user specifies the end point grid numbers of the first member in the string and the number of members defined by the string. The increment between the user supplied grid point numbers for the first member of the string is assumed to be the same for all members in the string. Members are defined to run from the first grid point of the string to the second, from the second to the third, from the third to the fourth, etc. Thus members have a beginning grid point and a terminating grid point as shown in figure 2.

Data for the MEMBER command takes the form,

MEMBER,IMEMB,igridA,igridB,ne,matname

where,

IMEMB is the user defined member number for the first member in the string

igridA is the grid point integer number at one end of the first member in the string that is being defined

igridB is the grid point integer number at the other end of the first member in the string that is being defined

ne is the number of members in the string being defined

matname is the number or alphanumeric name of a material property valid for all members in the string being defined; material properties are given by the MATPROP command

Members run from end A to end B.

NOTE 5.6-1: Member numbers in a string are assigned sequentially. If IMEMB is zero or blank, the first member in the string is automatically assigned a member number based on the order in which the MEMBER command appeared relative to other MEMBER commands.

Example 5.6-1: Defining a string of members

```
MEMBER,,10,20,10,2$ FIRST STRING OF TEN MEMBERS
40,110,120,2,3$ SECOND STRING OF TWO MEMBERS
```

In this example, two strings of members are defined. The first line of data is a MEMBER command defining a string of 10 beam members. The first member runs from grid point 10 to 20, the second from 20 to 30,...,the tenth from 90 to 100. Material properties of all these members are defined by the MATPROP command number 2. Since IMEMB is blank and this is the first MEMBER command in the data set,the members are automatically numbered from 1 to 10. The second line is also a MEMBER command. It defines a string of 2 members, the first running from grid point 110 to 120 and the second from grid point 120 to 130 with material properties defined by MATPROP command number 3. These members are numbered 40 and 41. If the associated MATPROP commands were those given in Examples 5.4-1 and 5.4-2, then the first string of members are fully flexible and the second string of members are fully rigid.

A condition label cannot be used in conjunction with a MEMBER command.

NOTE 5.6-2: There is no limit on the number of MEMBER commands that can be used, but the total number of members defined by all such commands cannot exceed the maximum value specified by the PORDER command.

NOTE 5.6-3: If an attempt is made to redefine an already defined MEMBER, the program will terminate at the pre-processor level and a fatal message will appear in the output file.

5.6.2 - Local Coordinate Systems

In addition to the stationary global axis system, the user should be aware of the rotating local axis systems. Local axis systems (X',Y'),see figure 5.5-1, are not defined by the user, but are automatically generated by the program. A local axis system is generated for each user defined beam member. This is accomplished internally by first establishing a direction for the member. In figure 5.5-1, this direction lies along a straight line from end A to end B of the member and is denoted by a unit vector V. Thus V changes its orientation as the member moves in space.

At any instant in time V makes an angle, denoted as θ, with the positive global X axis. The program then automatically defines local cartesian axes whose origin coincides with that of the global axes, but rotate, (not translate), with the vector. These axes are denoted as "local axes" or "local coordinate system" since they are different for each member. The local X' axis is always parallel to the local member vector and thus runs parallel to the member. The local Y' axis is normal to the member and makes the same angle with the global Y axis as the local X' axis makes with the global X axis.

Section 5.7 - Geometry Scaling

The geometry of the entire structure may be scaled and/or rotated through the use of an optional SCALE command. This command is useful when parameter studies are being performed and the user desires an easy way to vary structural size or orientation. Data takes the form,

SCALE,scalex,scaley rotate

where,

- scalex is a scale factor for the global x-direction; default is 1
- scaley is a scale factor for the global y-direction; default is 1
- rotate is an angle in radians for rotating the entire structure (positive angle measures rotation from the positive global x axis towards the positive global y axis); default is 0

The order of operations is to scale the structure in the global x-direction, the global y-direction, and then rotate the structure. If "scalex" or "scaley" are zero or blank, a default value of 1 is used for the associated scaling in that direction. If "rotate" is blank, a default value of zero is used. If a SCALE command is not used, no scaling of geometry nor rotation occurs.

Example 5.7-1: Scaling and rotating the structure

```
SCALE, 10.,,0.8$ SCALE X-DIRECTION BY A FACTOR OF 10
$ AND ROTATE STRUCTURE 0.8 RADIANS
```

In this example, the geometry of the entire structure is expanded by a factor of ten in the global x-direction and then rotated by a positive angle of 0.8 radians.

A condition label cannot be used in conjunction with a SCALE command.

Section 5.8 - Spring Member Definition

Optional linear springs are defined using the command SPRING. Springs may be either extensional or rotational and act in either a fixed direction or a variable direction. Data takes the form,

SPRING,igrid1,igrid2,idir,sprngk,psspng ? Cj

where,

igrid1 is the grid point at one end of the spring

igrid2 is the grid point at the other end of the spring

idir is the direction in which the spring acts, that is,

- idir=0, spring acts in a direction parallel to a straight line joining its end grid points and thus its line of action varies
- idir=1, spring acts in fixed x direction
- idir=2, spring acts in fixed y direction
- idir=3, spring acts in rotational direction

sprngk is the spring constant

psspng relates to the initial stretch or rotation in the spring; see equations below

Cj is the j th optional condition label which is defined with a Cj command; see section 5.15 for further details.

NOTE 5.8-1: The spring force at igrid1 is given by,

```
sprngk x [displacement(igird2)
                    - displacement(igrid1) + pssprng]
```

The spring force at igrid2 is opposite in sign to that at igrid1.

The spring torque if dir=3 at igrid1 is given by,

```
sprngk x [rotation(igrid2)
                  - rotation(igrid1) + pssprng]
```

The spring torque at igrid2 will be opposite in sign to that at igrid1.

Example 5.8-1: Defining a rotational spring

```
SPRING,10,110,3,1. E3, 0.1$ ROTATIONAL SPRING
```

In this example a rotational spring with spring constant of 1000 and initial relative rotation of 0.1 radians, acts between grids points 10 and 110.

NOTE 5.8-1: Up to 99 springs may be defined.

Section 5.9 - Damper Member Definition

Optional linear viscous dampers are defined using the command DAMPER. The damper may either extensional or rotational and either act in a fixed direction or a variable direction. Up to 99 dampers may be defined. Data takes the form,

DAMPER,igrid1,igrid2,idir,cdamp ? Cj

where,

igrid1 is the grid point number at one end of the damper

igrid2 is the grid point number at the other end of the damper

idir is the direction in which the damper acts and is defined exactly as that for a spring member, (see Section 5.8)

cdamp is the linear viscous damper constant

Cj is the j th optional condition label which is defined with a Cj command; see section 5.15 for further details

Example 5.9-1: Defining a linear viscous damper

```
DAMPER,10,110,1,100.$ DAMPER NUMBER 10
```

In this example, a damper with damping constant 100 joins grid points 10 and 110 oriented in the x direction

Section 5.10 - Optional Additions to Inertial Coupling And Lumped Masses

Inertial coupling of degrees-of-freedom is a natural consequence of the consistent mass approach used in the program, however the user may wish to specify additional inertial coupling. This is acccomplished via optional ADMASS commands. ADMASS commands may also be used to define lumped masses at any grid point. Data takes the form,

ADMASS,igrid1,idir1,igrid2,idir2,mass ? Cj

where, igrid1 and idir1 provide one degree-of-freedom of the structure and igrid2 and idir2 provide a second degreee-of-freedom of the structure; that is,

igrid1 is a grid point for the first degree-of-freedom

idir1 is the direction for the first degree-of-freedom

igrid2 is a grid point for the second degree-of-freedom; for defining lumped masses, node1 and node2 should be the same

idir2 is the direction for the second degree-of-freedom; for defining a lumped mass, idir1 and idir2 are generally the same

mass is the magnitude of the additional inertial coupling term between the two degrees-of-freedom

Cj is the j the optional condition label which is defined by a Cj command; see section 5.15 for further details

Example 5.10-1: Adding mass coupling and lumped mass

```
ADMASS,100,2,90,2,1.-2$ ADD OFF DIAGONAL INERTIA
90,2,100,2,1.E-2$ ADD OFF DIAGONAL INERTIA
110,1,110,1,5.E-2$ ADD LUMPED MASS
110,2,110,2,5.E-2$ADD LUMPED MASS
```

In this example, the first ADMASS command results in an additional inertial coupling with magnitude 0.01 specified to act between the degree-of-freedom indicated by grid point 100 and direction 2, and grid point 90 and direction 2. This will result in a value of 0.01 added to the appropriate off-diagonal term of the structural mass matrix. Though not required, the second ADMASS command retains the symmetry of the mass matrix. The third and fourth ADMASS commands are used to add a lumped mass of 0.05 at grid point 110; that is, a diagonal term in the appropriate locations of the mass matrix.

NOTE 5.10-1: Up to 99 ADMASS commands may be used.

NOTE 5.10-2: This optional command permits the inclusion of incompressible fluid effects wherein the presence of the fluid gives rise to an additional full mass matrix.

Section 5.11 - Beam Member Preload

An axial prestress may be applied to a fully flexible beam member by use of the optional PRELOAD command. Data takes the form,

PRELOAD,m,preload

where,

m is a number of a fully flexible beam member; i.e., MATPROP command asociated with this member has MTYPE=0

preload is an initial member load, positive for tension and negative for compression.

Example 5.11-1: Specifying member preload

```
MEMBER,1,6,7,5,2$ FIVE MEMBERS DEFINED
      .
      .
      .
PRELOAD,1,1000.$ PRELOAD IN MEMBER 1
2,2000.$ PRELOAD IN MEMBER 2
3,2000.$ PRELOAD IN MEMBER 3
4,-500.$ PRELOAD IN MEMBER 4
5,-1000.$ PRELOAD IN MEMBER 5
```

In this example, member number 1 which joins grid points 6 and 7 has a tensile preload of 1000, member 2 has a preload of 2000, member 3 has a preload of 2000, member 4 has a preload of -500 and member 5 has a preload of -1000.

A condition label cannot be used in conjunction a PRELOAD command.

NOTE 5.11-1: Members for which no PRELOAD command is provided are assumed to have a zero pre-load.

Section 5.12 - Externally Applied Loads and Gravity

Externally applied loads may be imposed at any grid point and defined to act in any direction in the global coordinate system.

The time variation of the loads may be specified in a piecewise linear form through the use of a user supplied table of values. In addition, options are provided for loads which are constant or vary sinusoidally with time.

Loads are defined in terms of load sets. A load set consists of a single time varying load with a particular magnitude and direction applied at a set of grid points. The following command is required for each set of applied loads.

APPLOAD - to identify a load set and to indicate how, where, when and under what conditions, if any, a specified time varying load set is applied; an APPLOAD command is required when any load set is defined

The APPLOAD command takes the form,

APPLOAD,nload,imemb,dir,scale,on,off,igrid1,igrid2,...? Cj

where,

nload is an integer load identifcation number

imemb is the identification number of the member whose local coordinate system is used to define the direction of the applied load; if imemb is zero or left blank, the global system is used

alpha=direction of load in radians as measured from
either the
global x axis (positive angles run from positive x to positive y axes) if imemb =0
or the
local x' axis (positive angles run from positive x' to positive y' axes) of member imemb when imemb>0; see figure 5.18-1

alpha may be a user defined variable Qk, where Qk is the kth user defined variable, (see NOTE 5.12-1)

alpha=TORQUE for applied torques (positive torques act from positive global x to positive global y axis)

scale is a factor used to scale the load magnitude values given by the LOAD, LOADSIN or LOADTIM commands defined subsequently and if zero or left blank, default value of 1 is used; scale may be a user defined variable Qk, where Qk is the k th user defined variable (see NOTE 5.12-2)

on and off are the absolute values of time at which the load set is applied and removed, respectively

igrid1,igrid2,... are the grid points at which the load set is to be applied or Ln where n is the identifier of a defined list (see page 5.2-1)

Cj is the j th optional condition label which is defined with a Cj command; see section 5.15 for further details

NOTE:5.12-1: "Alpha" may be specified via a user defined variable, Qk, where Qk is the k th user defined variable as described in section 5.16. Qk may be defined as a FORTRAN expression which depends on variables of the structural response such as velocity, acceleration, internal forces, etc. For example, follower forces can be specified by defining "alpha" as a fixed angle between a rotating beam member and the applied force. (See example 5.16-4.) Table 4.4-1 provides a complete list of admissible variables for Qk definitions.

NOTE 5.12-2: "Scale" may be specified via a user defined variable, Qk, where Qk is the k th user defined variable as described in section 5.16. Qk may be defined as a FORTRAN expression which depends on variables of the structural response such as velocity, acceleration, internal forces, etc. For example, this permits the user to define control forces with or without time delay. (See examples 5.16-2, 5.16-5 and 5.16-6.) Table 4.4-1 provides a complete list of admissible variables for Qk definitions.

Example 5.12-1: APPLOAD Command for Applied Forces

```
DEFINE:@6=1.5707963
APPLOAD,1,,@6,5.0,0.0,40.0,10,15,20$ APPLIED LOAD AT
$ GRID POINTS 10, 15 AND 20
```

In this example, an applied load set acts at grid points 10, 15 and 20. The load direction is prescribed to be 90 degrees from the global x axis (or parallel to the global y axis). The scale factor is 5 and the load set is applied at time 0 and removed at a time of 40 units. The first entry of the command identifies an additional command which identifies the time variation of the load set which is described subsequently.

Example 5.12-2: APPLOAD Command for Applied Torque

```
APPLOAD,2,,TORQUE,4.5,3.0,12.0,10,15,20$ TORQUE AT
$GRID POINTS 10, 15 AND 20
```

In this example, an applied load set consists of torques at grid points 10, 15 and 20. The scale factor is 4.50 and the load set is applied at 3 units of time and removed at 12 units of time. The first entry of the command identifies an additional command which identifies the time variation of the load set which is described subsequently.

Any of the following commands may be used to prescribe the time variation of applied loads.

LOAD - to specify a constant load in time (or a system response dependent load when a user defined Qk variable is employed as a scale factor in the APPLOAD command; see NOTE 5.12-2

LOADSIN - to specify a sinusoidally time varying load

LOADTIM - to specify a piecewise linearly time varying load

NOTE 5.12-3: Each of these commands may be referenced by many APPLOAD commands. The APPLOAD commands take the time variation specified in these commands and scale, orient and place them at prescribed grid points on the structure. If these commands are not referenced by APPLOAD commands, they are inactive.

The optional LOAD command takes the form,

LOAD,nload,amp

where,

nload is a unique load identification number which is referenced by APPLOAD commands

amp defines the load value

Example 5.12-3: Specifing a constant load in time

```
DEFINE: @6=1.5707963 $PI/2 DEFINED
APPLOAD,1,,@6,5.0,0.0,40.0,10,15,20$ APPLIED LOAD AT
$ GRID POINTS 10, 15 AND 20
LOAD:1,20.$ LOAD OF 20 APPLIED
```

In this example, a constant load is referenced by an APPLOAD command. Thus the load acts in a direction 90 degrees from the positive global x axis; that is, along the positive y axis and acts at grid points 10, 15 and 20 from time 0 to time of 40 units.. From the LOAD command, it has an amplitude of 20. The load amplitude is scaled by a factor of 5 to produce a total load amplitude of 100.

A condition label cannot be used in conjunction with a LOAD command, but can be used in with an APPLOAD command which references it.

The optional LOADSIN command takes the form,

LOADSIN,nload,freq,phase,amp

where,

nload is a unique load identification number which is referenced by APPLOAD commands

freq is the frequency of the load in hertz

phase is the phase angle of the sinusoidal load in radians

amp is the amplitude of the sinusoidal load

Use of LOADSIN will then result in a sinusoidal load from time=tloadon to tloadoff namely,

$$(amp)\ \sin[2\pi(freq)(t-tloadon) + phase]$$

where tloadon and tloadoff are prescribed in the APPLOAD command which references the LOADSIN command through the use of the LOADSIN identification number.

Example 5.12-4: Sinuoidal Time Variation of Torque

```
APPLOAD,2,,TORQUE,4.5,0.0,12.0,10,15,20$ TORQUE AT
$ GRID POINTS 10, 15 AND 20
LOADSIN,2,2.0,0.0,3.0$ SINUSOIDAL LOAD IN TIME
```

In this example, a sinusoidal load is referenced by an APPLOAD command. Thus a torque acts at grid points 10, 15 and 20 from time 0 to time of 12 units. From the LOADSIN command, it has a 2 Hertz variation, no phase shift and an amplitude of 3. The amplitude will be scaled by 4.5 to yield an amplitude of 13.5.

Example 5.12-5: Cosine Variation of Load

```
DEFINE,@1=3.14159265359$ PI DEFINED
DEFINE,@2=0.7853982$ PI/4 DEFINED
APPLOAD,9,,@2,5.0,0.0,40.0,10,15,20$ APPLIED LOAD AT
$ GRID POINTS 10, 15 AND 20
        .
        .
        .
LOADSIN,9,2.0,@1,3.0$COSINE VARIATION IN TIME
```

In this example, the value of π is defined somewhere in the data set. A sinusoidal load associated with the first APPLOAD command, having a frequency of 2 Hertz, phase shift of π radians and amplitude of 3 is defined. Thus a load with cosine type variation is indicated. From the APPLOAD command the time duration is from time 0 to a time of 40 units and the load acts in a direction given by @6 or 45 degrees from the positive global x axis and is applied at grid points 10, 15 and 20.

The LOADSIN command cannot be used in conjunction with a condition label, but an APPLOAD command which references it can have a condition label.

The optional LOADTIM command takes the form,

```
LOADTIM,nload,tt,k,time1,load1,time2,load2,...,timek,loadk
```

where,

nload is a unique identification integer which is referenced by APPLOAD commands

tt is the time type, either ABS, if time values used in this command are absolute or REL if time is measured relative to the time the load is applied

k is the number of pairs of (time,load) values used to define a piecewise linear time variation

time1,load1 is the first pair of (time,load) entries

time2,load2 is the second pair of (time,load) entries

timek,loadk is the k th pair of time load entries

Time values, time1, time2, etc., must be given in ascending order. Linear interpolation is used between the time,load pairs and linear extrapolation is used outside the range of the time,load pairs.

Example 5.12-6: Piecewise Linear Load in Time

```
APPLOAD,4,,TORQUE,4.5,0.0,100.0,10,15,20$ TORQUE AT
$GRID POINTS 10, 15 AND 20
LOADTIM,4,6,ABS, 65.0,2.0, 10.0,2.0, 15.0,4.0, &
20.0,4.0, 100.0,0.0, 1000.0,0.0 $ LINEAR PIECEWISE
$VARIATION
```

In this example, a piecewise linear load variation in time is to be applied in conjunction with instructions in an APPLOAD command. Thus the load set consists of torques acting at grid points 10, 15 and 20. From the LOADTIM command, six pairs of time/load values are given to define the piecewise variation of the load. Linear interpolation is used between the pairs of values and linear extrapolation is used outside the range of the values. Thus, for instance, the load value at time 0 is 2 and that at time 2000 is 0.

A condition label cannot be used in conjunction with a LOADTIM command, but an APPLOAD command which references it can have a condition label.

The optional GRAVITY command uses the well known acceleration at earth's sea level to set the gravity constant in Newton's Gravitational Law. Gravity forces are applied to all grid points. To account for variations in gravity force throughout the structure and thus gravity gradient, the coordinates of the earth's center in the global axis system is required as input. Data takes the form,

```
GRAVITY,g,xearth,yearth
```

where,

g is the acceleration due to gravity at sea level

xearth is the global x axis location of the earth's center

yearth is the global y axis location of the earth's center

Example 5.12-7: Specifying a Gravity Field

```
GRAVITY,32.2,0.0,-2.11e6$ GRAVITY FIELD DEFINED
```

In this example, the units being used in the problem are feet and seconds. The acceleration due to gravity is 32.2 feet per second squared and the coordinates of the earth's center in the global axis system are x=0 and y=-2.11e7 feet (or 3996 miles). The gravity load is applied to all grid points.

A condition label cannot be used in conjunction with a GRAVITY command.

NOTE 5.12-4: Since the GRAVITY command places gravity forces at all grid points and orients them toward the earth's center, an APPLOAD command is not used to reference a GRAVITY command.

NOTE 5.12-5: The GRAVITY command is not presently available. However, gravity can be approximated through the use of SPRING commands as done in the following example.

Example 5.12-8: Gravity Load Applied Through Use of SPRING Commands

```
SPRING:100,1,0,-@3,@4 $SPRING TO SIMULATE GRAVITY $
AT GRID POINT 1
```

In this example, the earth's center is placed at grid point 100. The spring acts on a straight line joining grid point 100 and grid point 1 which is on the structure. The spring has an inital stretch which simulates the gravity force prior to structure deformation and motion, and a spring constant which simulates gravity gradient. The initial stretch and spring constant are specified with DEFINE commands using the formulas:

psspng = (initial stretch)/ (spring constant)

$$= [gm(R_e/R_o)^2]/[-2gm(R_e/R_o)^2/R_o] = -R_o/2$$

sprngk = spring constant = $-2gm(R_e/R_o)^2/R_o$

where g is the acceleration due to gravity at sea level, m is a quantity of mass lumped at the grid point to simulate the mass of all members and additional mass attached to that grid point, R_o is the radial distance from the earth's center to the initial position of the grid point and R_e is the earth's radius.

These formulas represent a first approximation to gravity gradient phenomena.

Section 5.13 - Constraints

The following types of constraints are currently available:

(1) single degree-of-freedom constraints
(2) linear multi-degree-of-freedom constraints
(3) pin connection between two grid points
(4) rigid member constraints
(5) inequality constraints

5.13.1 - SDFC (Single Degree-of-Freedom Constraint) Command

Single degree-of-freedom constraints enforce the motion at a specified grid point and direction to be a given fixed value through the use of a SDFC command. Data takes the form,

SDFC,igrid,idir,rhs ? Cj

where,

igrid is the constrained grid point number

idir is the contrained direction (idir=1,2 or 3 constrains motion in the global x, y or rotational direction, respectively)

rhs is the constant value to which the constrained degree-of-freedom is set

Cj is the j th optional condition label which is defined with a CL command; see section 5.15 for further details

Example 5.13-1: Specifying a single degree-of-freedom constraint

```
SDFC,10, 1 ,0.0$ CONSTRAINT AT GRID POINT 10
```

In this example, grid point 10 is constrained to zero in the direction parallel to the global x axis.

Note 5.13-1: The grid point number and direction specifications of the SDFC command are translated by the program to a particular problem degree-of-freedom. This degree-of-freedom, unless otherwise specified through the use of a INDDOF command, will be treated by the program as a dependent motion and will be eliminated from the problem. (Section 5.13.6 describes the use of the the INDDOF command.) The reaction associated with the degree-of-freedom may be recovered for printing through the use of the REACT command described in Section 5.18. Reactions may also be used in condition statements or Q-variables as described in sections 5.15 or 5.16, respectivley.

5.13.2 - Multi-Degree-Of-Freedom Constraints

Multi-degree-of-freedom commands enforce a linear relationship between a collection of structural degrees-of-freedom. This is accomplished through the use of the optional MDFC command. Data takes the form,

MDFC,k,igrid1,idir1,coef1,igrid2,idir2,coef2,...,rhs?Cj

where,

k is the total number of degrees of freedom appearing in the MDFC command

igrid1 is the grid point number of the first degree-of-freedom in this command

idir1 is the direction of the first degree-of-freedom in this command; idir1=1,2 or 3 for direction parallel to global x axis, parallel to global y axis or rotation, respectively

coef1 is the coefficient for the first degree-of-freedom in this command

igrid2 is the grid point number of the second degree-of-freedom in this command

idir2 is the direction of the second degree-of-freedom in this command; idir2=1,2 or 3 for direction parallel to global x axis, parallel to global y axis or rotation, respectively

coef2 is the coefficient for the second degree-of-freedom in this command

rhs is the right hand side of the linear multi-degree-of-freedom command

Cj is the j th optional condition label which is defined with a Cj command; see section 5.15 for further details

The linear multi-degree-of-freedom constraint takes the equation form,

$$(coef1)(dof1) + (coef2)(dof2) + \ldots + (coefn)(dofm) = rhs$$

if Cj is TRUE

where,

dof1, dof2,...,dofm are the degrees-of-freedom associated with a grid point and direction appearing in the MDFC command

Example 5.13-2: Multi-Degree-of-Freedom Constraint Involving Three Degrees-of-Freedom

```
MDFC,3,30,2,1.0,35,2,1.0,40,2,-1.0,0.0
```

In this example, the motion of grid point 30 in the global y direction plus the motion of grid point 35 in the global y direction minus the motion of grid point 40 in the global y direction is constrained to be zero.

Example 5.13-3: Single-Degree-of-Freedom Constraint

```
MDFC,1,10, 1,1.0,0.0$CONSTRAINT AT GRID POINT 10
```

In this example, grid point 10 is constrained to zero in the direction parallel to the global x axis.

NOTE 5.13-2: Grid point number, direction and axis system specifications are translated by the program to a particular problem degree-of-freedom. As the program processes the problem constraints, it automatically selects a dependent degree-of-freedom from each constraint equation when that constraint is processed. In turn that dependent degree-of-freedom is automatically eliminated from all currently unprocessed constraints and from the system equations of motion. When a constraint is processed, the dependent degree-of-freedom selected is that currently having the largest coefficient in the constraint equation, but not one which the user has declared to an independent degree-of-freedom through an INDDOF command. However, the coefficients of the constraints generally change during the constraint processing procedure and so the user cannot usually predict which degree-of-freedom will be treated as dependent in an MDFC command prior to program execution. The user will know though, that degrees-of -freedom specified in an INDDOF command are independent degrees-of-freedom and will not be eliminated from the problem. (See section 5.13.6 for further details on use of the INDDOF command.)

Note 5.13-3: The reaction force associated with the dependent degree-of-freedom of an MDFC constraint can be recovered through the use of a REACT command as described in section 5.14, however, the user will usually not know a priori which degree-of-freedom has been treated as dependent. Of course when only one degree-of-freedom appears in the MDFC command it necessarily becomes a dependent degree-of-freedom, unless it is listed in an INDDOF command. See Section 5.13.6 for further details on use of INDDOF commands.

NOTE 5.13-4: It is quite possible that some constraint equations may not be independent of other constraint equations. The program will automatically recognize and handle such a condition. The user will also be informed of its occurence.

5.13.3 - Pin Connected Grid Points

Pin connections between grid points can be accomplished by the use of two MDFC commands which would constrain the motions of the two points parallel to the X and Y global axes to be identical. If many pin connections exist in a structure, the composition of the required MDFC commands can become tedious. To expedite the input of pin connection specifcations, an optional PIN command may be used. This command results in the automatic generation of the appropriate MDFC commands. The PIN command takes the following form,

```
PIN,igrid1,igrid2 ? Cj
```

where,

igrid1 and igrid2 are the grid point numbers where the pin connection exists

Cj is the j th optional condition label which is defined by a Cj command; see section 5.15 for further details

Example 5.13-4: Pin Connection Using PIN Command

```
PIN,10,40$PIN CONNECTION BETWEEN GRID POINTS 10 AND
$ 40
```

In this example, a pin connection is specified between grid points 10 and 40. Consequently, the motions of the two points parallel to the global x and y axes are identical, but the grid points may rotate freely of one another.

5.13.4 - Rigid Member Constraints

Rigid members impose constraints between the motions of their end grid points. These constraints are automatically accounted for when rigid members are specified in a MATPROP command; see Section 5.4. The user need not concern himself with defining these constraints.

5.13.5 - Inequality Constraints

Inequality constraints may be applied by the user. Such constraints specify that a linear or nonlinear algebraic combination of variables must be less than or greater than some other linear or nonlinear combination of variables. Inequality constraints are applied through the use of condition labels. See for instance examples of section 5.15.

5.13.6 - User Selection of Independent Degrees-of-Freedom

In general, the program processes constraints by eliminating certain degrees-of-freedom from the problem. Such eliminated degrees-of-freedom become dependent degrees-of-freedom. These are selected by the program during execution and the user has little control over which degrees-of-freedom will be treated as dependent. However, the user can specify which degrees-of-freedom must be treated as independent by the program; that is, cannot be dependent degrees-of-freedom, through the use of the optional INDDOF command.

The INDDOF command takes the form,

```
INDDOF:k,igrid1,idir1,igrid2,idir2,...
```

where,

k is the number of degrees-of-freedom specified independent by this command

igrid1,igrid2,... are the grid point numbers associated with independent degrees-of-freedom

idir1,idir2,... are the directions in global axis system associated with independent degrees-of-freedom at grid points node1,node2,..., respectively

Example 5.13-5: Specification of independent degrees-of-freedom

```
INDDOF:3,10,1,4,2,25,3 $ INDEPENDENT DOF'S
```

In this example, three degrees-of-freedom are specified which the program must treat as independent. They are as follows:

- grid point 10 in direction parallel to global x axis
- grid point 4 in direction parallel to global y axis
- grid point 25 in rotational direction

A condition label cannot be used in conjunction with a INDDOF command.

Note 5.13-6: INDDOF commands override any other specifications so that an attempt to declare a constraint on degrees-of-freedom specified in an INDDOF command are ignored by the program. A warning message will be provided in the output file to indicate this occurrence.

Section 5.14 - Initialization

Initial displacements and or velocities in the global coordinate system may be defined by the user. Initial conditions on all degrees-of-freedom are assumed to be zero unless they are defined through the use of DIS and VEL commands. Data takes the form,

```
INIT,scaledis,scalevel
DIS,igrid,idir,coord,dis
VEL,igrid,idir,coord,vel
```

where,

scaledis and scalevel are scale factors for all initial displacements and velocities respectively; if the INIT command is not used, the default values are unity

igrid is the grid point of the degree-of-freedom to be initialized

idir is the direction of motion that is to be initialized; idir=1, 2 or 3 for motions parallel to the global x axis, parallel to the global y axis or positive rotations

coord is the identification number of a member in whose local coordinate system the initial condition(s) are described; if coord =0 or is left blank, the global system is implied

dis and vel are the initialized values of the displacement and velocity respectively

Example 5.14-1: Specification of initial conditions

```
DIS,20,1,,5.2
VEL,20,2,0,-1.0
```

In this example, the displacement at grid point 20 in the global x direction is initialized to 5.2 while the velocity of the same grid point in the global y direction is initialized to -1. Since no INIT command is used the scale factor default of one is assumed.

Note 5.14-1: The general philosophy used in the program is to allow the program to execute if at all reasonably possible. Especially is this philosophy important in the processing of initial conditions and constraints. (See also section .) In specifiying initial conditions it is quite likely that the user may impose conditions which are contradictory to user specified constraints, including rigid member constraints. In such a case, the problem constraints will override the user supplied initial conditions. The program will employ some of the user supplied initial conditions and calculate the others so that the constraints are satisfied. The program institutes a priority for deciding which user supplied initial conditions are employed and which are calculated from the constraints.

Priority is on a first appearance basis, so that when a contradiction between initial conditions and constraints occurs the program discards the last user supplied initial condition and checks to see if a contradicition still exists. If there is no longer a contradiction, program execution proceeds. If a contradiction still exists the next to last supplied initial condition is discarded and a contradiction check made again. The process continues until the contradiction is eliminated. (In some cases it may not be possible to eliminate the contradiction. For example, when only one non-zero initial condition exists and the degree of freedom it is associated with is constrained to zero.) Warning messages will appear in the output file indicating which initial conditions have been discarded and the results at the starting time will reveal the effect of the constraints on the initial conditions.

The user will find this procedure to have certain advantages. For example, if degrees-of-freedom are slaved to one another through constraints, the user need not unduly be concerned about what could be a very complicated kinematics problem in determining the appropriate complete and consistent set of initial conditions. The user may decide not to specify the initial conditions at those locations which are believed to be slaved, via the constraints, to other locations where initial conditions are specified. The program will first discard the defaulted initial conditions as it attempts to produce a well defined problem. The effect is that the kinematics of the problem are automatically taken care of by the program.

Section 5.15 - Condition Statements and Labels

Condition statements are optional logic expressions written by the user in FORTRAN-like syntax, which at any time step may be TRUE or FALSE. A condition statement is identified by a condition label. The j^{th} condition statement is specified through the use of a CLj (or Cj) command. The command takes the form,

CLj, FORTRAN expression

where j is an integer identification number.

Associated with the j^{th} condition statement is a condition label, denoted Cj. LATDYN commands are made conditional by attaching a question mark, "?", followed by a condition label Cj to the end of the command. If the condition statement associated with the condition label is TRUE then the data commands to which the condition label is attached are active. If the associated condition statement is FALSE then the data command to which the condition label is attached are inactive.

Condition labels may be attached to the following commands:

ADMASS
APPLOAD
DAMPER
FREQ
INCMASS
MATPROP
MDFC
PLOTINC
PRINT
SDFC
SPRING
STIFF
STOP
TIMSTEP
SET
RESET
KEEP
ROTLOCK

In general, the use of such conditions gives the user considerable flexibility in executing the program. For example, Cj commands may be used to construct inequality constraints by defining a unique condition label to be associated with an inequality condition statement written in FORTRAN and attaching the condition label to a constraint command.

Whenever condition labels activate or deactivate commands which can affect the mass and/or stiffness matrix, these matrices are automatically updated. A message in the output file will indicate this occurrence. Condition labels on the following commands can lead to updating of mass and/or stiffness matrices:

ADMASS (mass matix updated)
DAMPER (mass matix updated)
SPRING (stiffness matrix updated if STIFF command is used)
INCMASS (mass matrix updated)
MATPROP (mass matrix updated and stiffness matrix also updated if STIFF command is used)
MDFC and SDFC (mass matrix updated and stiffness matrix also updated if STIFF command is used)
STIFF (stiffness matrix updated)
TIMSTEP (mass matrix updated if DAMPER command is used or INTGTYP is 5

The logic expressions in the Cj command are written in FORTRAN-like syntax using a special set of variables and parameters related to the variables of the problem. The variables may be displacements, velocities, accelerations, internal and external forces, axial strains, member rotations, structural frequency and time.

The admissible variables in a condition statement are listed in Tables 4.4-1 and take the form,

XLOC(n)	for the current location of grid point n parallel to the global x axis
YLOC(n)	for the current location of grid point n parallel to the global y axis
ROT(m)	for the current rotated position of member m measured positive in a direction from the positive global x to the positive global y axis
X(p,n)	for component of motion quantity (displacement, if p=D; velocity, if p=V; acceleration, if p=A) at grid point n parallel to the global x axis
Y(p,n)	for component of motion quantity (displacement, if p=D; velocity, if p=V; acceleration, if p=A) at grid point n parallel to the global y axis

XP(p,n,l)	for component of motion quantity (displacement, if p=D; velocity, if p=V; acceleration, if p=A) at grid point n parallel to a local x'axis (parallel to a designated member, l)
YP(p,n,l)	for component of motion quantity (displacement, if p=D; velocity, if p=V; acceleration, if p=A) at grid point n parallel to a local y'axis (transverse to designated member, l)
PHI(n)	for rotation at grid point n in the positive angular direction; from positive global x axis to positive global y axis
MAG(p,n)	for magnitude of motion quantity (displacement, if p=D; velocity, if p=V; acceleration, if p=A) at grid point n
ANGLE(p,n)	for angular orientation of motion quantity (displacement, if p=D; velocity, if p=V; acceleration, if p=A) at grid point n; positive angles are measured from positive global x axis towards positive global y axis
AX(m)	for axial force in designated member m (positive for tension)
SHA(m), SHB(m)	for shear force at end A or B, respectively, of beam member m
BM(m,iend)	for bending moment at A, iend=0, or end B, iend =1, of beam member m
FEXTX(j), FEXTY(j)	for component of j^{th} external force parallel to the global x or y axis, respectively
FEXTM(j)	for the magnitude of the j^{th} external force
FEXTA(j)	for angular orientation of the j^{th} external force; positive angles are measured from the positive global x to positive global y axis
STRAIN(m)	for axial strain in member m

REACTX(n)	for reaction force at grid point n in the global x-direction
REACTY(n)	for reaction force at grid point n in the global x-direction
REACTT(n)	for reaction torque at grid point n in the global x-direction
REACTM(n)	for the magnitude of the reaction force at grid point n
REACTA(n)	for the angular orientation of the angular force at grid point n
REACTC(n,l)	for the component of the reaction force at grid point n parallel to the local coordinate system of member l
T	for time
NTSTEP	for time step number
OMEGA(k)	for frequency in radians per unit time of designated mode k, where the absolute value of k designates the k th lowest mode if k is negative and the k th highest mode if k is positive
CLj	for the j^{th} logical condition; provides logical TRUE or FALSE for condition label Cj

where,

p is a parameter indicating the type of motion desired as,

D, for displacement

V, for velocity

A, for acceleration

l designates the member number associated with a local axis system

m is a member number

n is a grid point number

The defined variables may be combined as logical expressions written in FORTRAN using the following logical operators:

.EQ. - equal to
.GT. - greater than
.LT. - less than
.GE. - greater than or equal to
.LE. - less tha or equal to
.AND. - logical "and"
.OR. - logical inclusive "or"

In addition to the variables defined above, the following constants may also be used:

PI for π, PIH for $\pi/2$, PIQ for $\pi/4$ and PI2 for 2π.

The program uses the condition statement to automatically create a FORTRAN subroutine. The subroutine employs the user given algebraic expressions of the condition label exactly as prescribed. Any legal FORTRAN operation is available to the user in writing the algebraic expressions of the condition labels. For example, the algebraic expressions may make use of the intrinsic FORTRAN in-line functions ABS, IABS, SQRT, SIN, COS, TAN, etc. If FORTRAN errors exist in the condition label expressions, compiler errors will result. Hence if compilation errors arise it is suggested that the user first examine all FORTRAN input expressions.

Example 5.15-1: Condition statement used to construct a nonlinear spring

```
CL1, X(D,6)  .GT.  X(D,4)   $ DEFINES C1
     Δ                Δ
     |                |
start of FORTRAN expression   end of FORTRAN expression
```

In this example, the condition statement to be associated with a condition label C1 is defined. The condition is true when the displacement in the global x direction at grid point 6 is greater than the displacement in the global x direction at grid point 4. This condition statement may be applied to any of the data commands listed above by simply attaching a question mark, "?", followed by the condition label C1 to the right hand end of the data command such as,

```
SPRING,4,6,1,10.0,0.0 ? C1 $SPRING WITH CONDITION
$ LABEL
```

This command defines a spring between grid points 4 and 6 which acts in the global x direction only when condition C1 is true. Effectively a nonlinear spring which can only take tensile strains is defined.

NOTE 5.15-1: In many problems it is useful to have multiple commands of a single type some of which may have attached condition labels. The use of multiple occurrences of TIMSTEP, INCMASS, STIFF, FREQ, MATPROP, PRINT AND PLOTINC, allows the user to vary the data supplied by these commands during the execution according to rules the user specifies through condition labels attached to these commands.

Which of the multiple commands is active at any time during execution depends upon the sequence in which the commands appear in the data set and whether their attached labels are TRUE or FALSE. Throughout the manual this is often referred to as condition-order priority.

Condition-order priority implies that the program examines the last command of such a multiple set. If it has no attached condition label, then it is always active, whereas if it has a condition label, it is active only if the condition is TRUE. If the condition is FALSE then the program examines the preceeding command of the multiple set in the same fashion. The process continues until an active command is found or all commands in the multiple set are exhausted. Thus, the last command of a multiple set with a condition label has priority if the condition is TRUE. (See subsequent examples 5.15-2, 5.15-7, 5.15-8 and 5.15-9.) For convenience in using this document, condition-order priority is defined wherever commands which depend on this process appear in the text.

Example 5.15-2: Condition statement used to change time step during job execution

```
CL2, T .GE. 5.$DEFINES C2
TIMSTEP,.01 $ INITIAL VALUE OF TIME STEP
TIMSTEP,.001? C2$ DECREASED TIME STEP WHEN C2 TRUE
```

In this example, multiple TIMSTEP commands are used. Because the order of their appearance in the data set is important, no contradictory instructions to the program result. A condition statement with label C2 is defined and then applied to the second TIMSTEP command. The first TIMSTEP command sets the time step to 0.01, however, the subsequent TIMSTEP command when active, will supersede it. Thus, when 5 time units are reached, condition C2 is TRUE, the second TIMSTEP command is activated and the time step is changed to 0.001. Effectively, the variation of the solution time step is thereby controlled by the user.

Example 5.15-3: Inequality constraint constructed with condition statement and single-degree-of-freedom constraint

```
CL3, Y(D,10) GE. 2.$DEFINES C3
SDFC,10,2,2.0 ? C3$ CONSTRAINT ACTIVE WHEN C3 TRUE
```

In this example, condition statement with label C3 is defined and then applied to an SDFC command. The displacement of grid point 10 parallel to the global y axis is constrained to be less than 2. If grid point 10 attempts to displace from its initial position more than 2 units parallel to the global y axis, a single degree-of-freedom constraint is applied which fixes the global y displacement at 2 units. Effectively, the user has constructed a capture mechanism at y=2 which captures grid point 10

Example 5.15-4: Inequality constraint for rebound behavior constructed from condition statement and single-degree-of-freedom constraint

```
CL4, Y(D & $ LINE ENDS WITH & SYMBOL THUS IS
$ CONTINUED ON NEXT LINE
,10) .GE. 2.0 .AND. REACTY(10) .LE. 0.0$ DEFINES C4
SDFC,10,2,2.0 ? C4$ CONDITIONAL CONSTRAINT AT GRID
$ POINT 10 IN Y DIRECTION
```

This example is similar to that of the previous example, 5.15-3. However, whereas in the previous example, grid point 10 was captured permanently by some mechanism at y=2, in this example, grid point 10 is only constrained at y=2 if the reaction force at grid point 10 is negative. Effectively, the user has constructed a wall at y=2 and allowed grid point 10 to rebound from that wall when the force between the grid point and the wall is pulling rather than pushing on the structure at that point.

To accomplish this rebound behavior it was necessary to use the admissible variable REACTY which recovers the reaction force at the constraint; that is, in the y direction at grid point 10. Prior to impact, C4 is FALSE since the displacement of grid point 10 parallel to the global y axis is less than 2 and the SDFC command is inactive. Consequently, the reaction force is zero before impact. When grid point 10 impacts the wall, the reaction force is still zero so that C4 will now be TRUE and the SDFC command will be activated. Thereafter, the reaction force will continue to be monitored. If the reaction force changes sign, C4 becomes false, the SDFC command will be deactivated and rebound will occur.

Example 5.15-5: Condition statement used to define early job termination

```
CL5,MAG(V,20) .GT. 1.E4 .AND. T .GT. 2.0$ DEFINES
$ C5
STOP? C5$ JOB TERMINATES WHEN C5 IS TRUE
```

In this example, a fifth condition statement is defined which is true when the magnitude of the velocity vector at grid point 20 exceeds 10,000 and time exceeds 2 units. The condition label is attached to a STOP command which will terminate execution if the fifth condition statement is true.

Example 5.15-6: Condition statement used to control mass matrix updating

```
INCMASS,100? C6$ MASS MATRIX UPDATE WHEN C6 IS TRUE

CL6,(PHI(V,1)+PHI(V,2)+PHI(V,3)+PHI(V,4)+PHI(V,5)&
+ PHI(V,6)+PHI(V,7)+PHI(V,8)+PHI(V,9) &
+ PHI(V,10))/10. .GT. 1000.$ C6 DEFINED
```

In this example, the mass matrix is not usually updated since an INCMASS command is not usually active. Only when the condition statement associated with label C6 is true, is the INCMASS command active. The condition statement depends on the average rotational velocity of the first ten grid points of the structure. For additional information on mass matrix updating see section 6.2.

Example 5.15-7: Condition statement used to designate application of applied force

```
APPLOAD,1,,0.75, 0.0,1.0, 30$ SCALE FACTOR IS
$ USUALLY ONE
APPLOAD,1,,0.75, 0.0,2.0, 30? C7$SCALE FACTOR TWO
$ WHEN C7 TRUE
LOADSIN,1, 10.0, 0.0, 15.2$ TYPE OF LOAD IS
$ SINUSOIDAL IN TIME
CL7,MAG(A,30) .LT. 1.E3$ DEFINES C7
```

In this example, a sinusoidal load in time is applied to grid point 30 with amplitude 15.2 in a direction 0.75 radians from the positive global x axis. However, when the acceleration at grid point 30 drops below a magnitude of 1000, the applied load scale factor is doubled; that is, the sinusoidal load amplitude becomes 30.4.

Example 5.15-8: Condition statement used to remove failed member

```
MEMBER,12,12,13,1,1,0$ THIS IS THE 12 TH DEFINED
$ MEMBER
MATPROP,1,10.5E6,.1,.01,2.588E-4$ NOMINAL MATERIAL
$ PROPERTY
MATPROP,1, 0.0, .1, .01, 0.0? C8$ FAILED MEMBER
$ PROPERTY
CL8, (ABS(AX(12)) .GT. 1.E4)$ DEFINES C8
```

In this example, the 12 th member is associated with the first material property. Two MATPROP commands are designated as material property number 1. The second appearing MATPROP command is associated with the removal of a member. The material property elastic modulus and density are made to vanish when the internal axial force of member 12 exceeds a failure value of 10,000 as defined by the eighth condition label. Thus the axial force determines the failure of the member and the member is effectively removed.

Example 5.15-9: Multiple condition statments used to remove failed member

```
MEMBER,12,12,13,1,1,0$ THIS IS THE 12 TH DEFINED
$ MEMBER
MATPROP,1,10.5E6,.1,.01,2.588E-4$ NOMINAL MATERIAL
$ PROPERTY
MATPROP,1, 0.0, .1, .01, 0.0? C8$ FAILED MEMBER
$ PROPERTY
CL8, (ABS(AX(12)) .GT. 1.E4)$ DEFINES C8
MATPROP,1,0.0,.1,.01,0.0? C9 $ FAILED MEMBER
$ PROPERTY WHEN TIME EXCEEDS 2
CL9,T .GT. 2.0$ DEFINES C9
```

This example is similar to 5.15-8 except that here an additional condition statement with label C9 is also used. Since the MATPROP command with the C9 label attached appears after that with the C8 label attached, the statement associated with C9 will take precedence. Thus the member stiffness will be reduced to 5.25E6 or half its nominal value after time exceeds two units whether or not the axial load has exceeded 10,000. Thus the failure condition with label C8 attached will have no relevance after a time of two units.

The above examples display some of the versatility in using condition labels such as in creating nonlinear members which act only in tension (or compression), controlling time steps, setting lock-up and rebound conditions, changing mass matrix update increment, instructing possible early job termination and removing failed members. They may also be used to apply control forces which depend on structural deformation quantities, contol printed and plotted output, to add mass to the structure due to impact with a foreign object and a host of other functions too numerous to list.

Section 5.16 - User Defined Variables (also denoted as Q-variables)

5.16.1 - General Discussion

In section 5.4 the command DEFINE was introduced. This allowed the user to define in a shorthand fashion, constants which often appear in the data set. In addition, the program allows the user to define variables which depend on the time varying quantities XLOC, YLOC, ROT, X, Y, XP, YP, PHI, MAG, ANGLE, AX, SHA, SHB, BM, FEXTX, FEXTY, FEXTM,FEXTA, OMEGA, STRAIN, NSTEP and T as defined in section 5.15. This enables the user to vary certain quantities appearing in data commands subject to the time varying quantities of the analysis. User defined variables may be used in any condition label, in a TIMSTEP command, in a MATPROP command and as a scale factor and/or load direction in APPLOAD commands. The use of user defined variables in TIMSTEP or MATPROP commands is not available in the current version.

5.16.2 Creating Q-Variables

User defined variables are prescribed with the optional SET and RESET commands.

To define a constant parameter Qn (which once calculated remains fixed), the SET command is used. Data takes the form,

SET,Qn=FORTRAN arithmetic expression ? Cj

where,

n is an identification integer

Cj is the j th optional condition label which is defined with a CLj command as discussed in section 5.15; Cj should not contain a user defined variable whose definition is itself subject to a condition label

The SET command creates a constant denoted as Qn where n is the identification integer of the SET command. Qn is initally set to zero. If the SET command has an attached condition, Qn is set to a fixed value equal to the arithmetic expression of the SET command when the condition first becomes TRUE. If no condition label is attached to the SET command, the arithmetic expression is evaluated at the first time step. Once the variable is set, its value remains fixed even if the condition label later becomes FALSE. The value of Qn defined by a SET command should not be changed by the use of another SET command or a RESET command.

Example 5.16-1: User variable as load scale factor via SET command

```
APPLOAD,12,,0.0,Q1,0.0,10.0,5 $ VARIABLE SCALE
$ FACTOR
LOAD,12,1.0$ SCALED LOAD
SET,Q1=(X(A,6) + X(A,7) + X(A,8))/3.0? C2 $DEFINES
$ Q1
C2,X(D,7) .GT. 0.5$ DEFINES C2
```

In this example, a variable Q1 is set to the average value of the accelerations parallel to the global x axis at three grid points when the condition label C2 first becomes true; that is, when the displacement in the global x direction of grid point 7 first exceeds one-half. Q1 is then used as a scale factor of an APPLOAD command which defines a load at grid point 5 over a maximum time interval of 0 to 10 time units. Since all user defined variables are automatically initialized to zero, the scale factor is initially zero, and thus the applied load, does not exist at all before C2 becomes TRUE. Since Q1 is defined with a SET command, its value is determined when the condition statement associated with label C2 becomes TRUE. The value of Q1 will not change thereafter even if the condition statement subsequently becomes FALSE.

To define a quantity whose value varies, the RESET command is used. It takes the form,

RESET,Qn=FORTRAN arithmetic expression ? Cj

where,

n is an identification integer

Cj is the j th optional condition label which is defined with a CLj command as discussed in section 5.15; Cj cannot contain a user defined variable whose definition itself is subject to a condition.

The RESET command defines a variable Qn, where n is the identification integer. At every time step for which the condition statement attached to the RESET command is true, the value of the arithmetic expression of the RESET command is calculated and the variable reset to that value. If no condition statement is attached, the variable is reset to the value of the arithmetic expression at every time step. The variable may depend on current or past values of the structural response. In this way RESET may be used to define control forces on the structure including time delay.

C-2

Example 5.16-2: User defined variable as load scale factor to yield velocity dependent control

```
DEFINE:@6=1.5707563 $ PI/2
RESET,Q2= YP(V,8)$ DEFINES Q2 PROPORTIONAL TO
$ VELOCITY
APPLOAD,3,,@6,Q2,0.0,10.0,8$ VARIABLE SCALE FACTOR
LOAD,3,206.5$ LOAD VALUE OF 206.5 BEFORE SCALING
```

In this example, Q2 is used as a scale factor for an applied load at grid point 8. From the RESET command, the scale factor, Q2, is equal to the velocity of grid point 8 parallel to the global y axis. The load amplitude of 206.5 in the third LOAD command, which is referenced by the APPLOAD command, will be multiplied by the variable scale factor. A velocity dependent control force is thus constructed. Unlike the previous example where the SET command was used, the RESET command permits the variable scale factor to continuously vary as the y-direction velocity of grid point 8 varies. To construct the same example with a time delay in the control force, see example 5.16-7.

Example 5.16-3: User variable in a condition statement

```
MEMBER,7,7,8,2,2$ DEFINE MEMBERS SEVEN AND EIGHT
RESET,Q3= YP(A,8,7) - YP(A,7,7) $DEFINES Q3 AS
$RELATIVE TRANSVERSE ACCELERATION OF TWO GRID
$POINTS COMPRISING MEMBER 7
RESET,Q4= YP(A,9,8) - YP(A,8,8) $DEFINES Q4 AS
$RELATIVE TRANSVERSE ACCELERATION OF TWO GRID POINTS
$ COMPRISING MEMBER 8
CL22, Q3 .GE. Q4$ DEFINES C22 IN TERMS OF Q3 AND Q4
INCMASS,1000 $MASS MATRIX UPDATE AT LEAST EVERY 1000
$TIME STEPS
INCMASS, 250? C22$ MASS MATRIX UPDATE MORE OFTEN IF
$C22 TRUE
```

In this example, beam members 7 and 8 are defined and the relative acceleration between their end points is set to user variables, Q3 and Q4. These variables then constitute a condition statement, CL22, which governs mass matrix updating.

Example 5.16-4: Combining user defined variables

```
RESET,Q3= YP(A,8,7) - YP(A,7,7) $DEFINES Q3 AS
$RELATIVE TRANSVERSE ACCELERATION OF TWO GRID
$POINTS COMPRISING MEMBER 7
RESET,Q4= YP(A,9,8) - YP(A,8,8) $DEFINES Q4 AS
$RELATIVE
$ TRANSVERSE ACCELERATION OF TWO GRID POINTS
$COMPRISING MEMBER 8
RESET,Q5=(Q3+Q4)/2.$AVERAGE OF Q3 AND Q4
APPLOAD,18,,Q5,,,16$APPLIED LOAD WITH VARIABLE
$MAGNITUDE AT GRID POINT 16
LOAD,18,1.0$THIS LOAD HAS A SCALE FACTOR OF Q5
```

In this example, variables Q3 and Q4 of example 5.16-3 are again used. They are averaged to produce Q5 which is employed as a scale factor for a load at grid point 16.

Example 5.16-5: Follower force using defined variable

```
MEMBER,14,9,10,1,ALUMINUM$ MEMBER DEFINED
RESET,Q9 = PHI(10)+PI/2$ DEFINE Q-VARIABLE NORMAL
$TO ROTATION OF MEMBER 14 AT GRID POINT 10
APPLOAD,3,Q9,3.0,,,10$ LOAD IN DIRECTION Q9 AT GRID
$ POINT 10 WITH MAGNITUDE 3
LOAD,3,1.$CONSTANT LOAD
```

In this example, a Q-variable is reset at every time step to an angular value given by the rotation of grid point 10 plus $\pi/2$. A constant in magnitude load is thereby oriented always normal to the beam members rigidly connected to grid point 10.

NOTE 5.16-7: Up to 99 user defined variables may be employed.

5.16.3 - Time Delayed Variables Via KEEP Command and Q-History Function

As already discussed, user defined variables, Qk, (k=1 through 99), are defined in SET or RESET commands. They depend upon the analysis variables listed in Section 5.15 and Table 4.4-1. During the program execution, Qk is assumed to be evaluated at the current time step. The value of Qk at previous values of time are generally destroyed. However, the program can be instructed to retain values of certain user defined variables at a number of previous values of time. This is accomplished with the use of the KEEP command. Data for the KEEP command takes the form,

KEEP: Qi,Qj,Qk,ns

where,

Qi, Qj, Qk are the ith, jth and kth user defined variables which may appear as arguments of QH functions

ns is the number of previous time values, before the current time, for which the values of Qi, Qj or Qk are to be retained in memory; the values of the Q variables will be saved at all ns previous time steps

Example 5.16-6: To retain previous values of a user defined variable

```
RESET, Q12 = AX(22)$ Q-VARIABLE EQUAL TO AXIAL FORCE
$ IN MEMBER 9
KEEP, Q9,1$ Q9 RETAINED FOR ONE PREVIOUS TIME STEP
```

In this example, a Q variable is defined to be the axial force in member 9. The value of Q9 is retained at one previous time step. Thus at any time during execution, the current value of Q9 and a retained value of Q9 one time step in the past is available to the user for employment in other commands.

If the user wishes to employ a retained value of a user defined variable, rather than the current value, in a command permitting user defined variables as data entries, then a Q-History (QH) function is used, as

QH(Qk,m)

where,

Qk is the k th user defined variable and appears in a KEEP command

m is an integer specifying the number of previous time steps at which Qk is to evaluated; only the magnitude of m is used by the program and the magnitude must lie between 1 and the value of ns in the KEEP command

QH functions may be used wherever a Qk variable is permitted.

Example 5.16-7: User defined variable as load scale factor to yield velocity dependent control with time delay

```
KEEP,Q2,5 $ PROGRAM TO RETAIN VALUES OF Q2 FOR
$ PREVIOUS FIVE TIME STEPS
RESET,Q2= YP(V,8)$ DEFINES Q2 PROPORTIONAL TO
$ VELOCITY
APPLOAD,3,,@6,QH(Q2,5),0.0,10.0,8$ VARIABLE SCALE
$ FACTOR
$ EVALUATED FIVE TIME STEPS PREVIOUS TO CURRENT TIME
LOAD,3,206.5$ LOAD VALUE OF 206.5 BEFORE SCALING
```

This example is similar to that of 5.16-2 except that a time delay equal to five time steps is used in calculating the control force.

NOTE 5.16-1: Only one KEEP command is permitted in a data set.

NOTE 5.16-2: The absolute value of m in the argument of the QH function must lie between 1 and the value of ns in the KEEP command. Since only the magnitude of m is used by the program it is recommended, as a memory aid to the user, that m be inserted as a negative integer in the argument of the QH function.

NOTE 5.16-3: Any or all of the defined Q-variables in the KEEP command may appear as arguments of QH functions.

Section 5.17 - Joint Lock-Up

5.17.1 - General Discussion

The user may desire to allow pin connected members to become rigidly connected when certain conditions are met. The grid points which make up the pin connection are denoted herein as the "joint" and the transition of the joint from pinned to rigid is denoted herein as "lock-up". Lock-up often occurs in analyzing the deployment of structures and in some mechanisms. A lock-up occurence involves an impact and consequently involves transfer of momentum across a joint. Such momentum transfer depends upon the properties of the locking joint and the members which come into the joint. The LOCKUP command is used to provide the properties of the joint when it locks-up. If the user does not wish to model the dynamics of the locking joint, the ROTLOCK command may be used instead. This command conserves the momentum of the entire structure without detailing the structural behavior in the neighborhood of the locking joint.

5.17.2 - LOCKUP Command

The joint model prescribed by the optional LOCKUP command consists of a nonlinear spring and viscous damper acting in parallel at the joint. The user provides the nonlinear spring and damper properties or uses certain default values. In addition, the user provides the condition under which lock-up is to occur in the form of a condition label.

When this condition is met, the nonlinear spring and damper are automatically activated. The spring and damper will quickly force the relative velocity between the grid points comprising the joint to zero. Once the relative velocity reaches zero, the spring and damper are automatically removed and replaced by a multi-degree-of-freedom constraint which enforces rigidity of the joint.

If no LOCKUP command is used, then the multi-degree-of-freedom constraint to enforce rigidity of the joint is applied as soon as the condition label defining lock-up is TRUE.

The nonlinear spring properties and the lock-up condition are provided by the user through the use of the optional LOCKUP command. Data takes the form,

LOCKUP,s1,s2,c1,c2,n1,n2,n3,n4,...?Cj

where,

s1 and s2 are coefficients describing the nonlinear lock-up spring constant, k, such that,

$$k = s1 + (s2)(Qj)^2$$

in which Qj is the current value of a user defined Variable; see below and section 5.15

c1 and c2 are coefficients describing a nonlinear joint damper whose damper value, c, is given by,

$$c = c1 + (c2)(Qj)^2$$

n1,n2,n3,n4,...is a string of grid point numbers comprising one or more joints; n1 and n2 comprise the first joint, n3 and n4 comprise the second joint, etc.

Cj is the j th condition label. As described in detail in section 5.15, the command to which the condition label is attached, in this case the LOCKUP command, is activated when the condition label is true. Thus, in this case, Cj expresses the condition which signifies joint lock-up. The condition label, Cj, is defined with the CL command as a logical expression written in FORTRAN. For use with the LOCKUP command, Cj must take the form,

Qj .GT. 0.0

where Qj is an arithmetic FORTRAN expression defined by the j th RESET command as described in section 5.16. The value of Qj may depend on current values of problem variables, such as, displacements, velocities or accelerations.

Example 5.17-1

```
LOCKUP,100.,0.0,10.0,0.0,11/12 ? C5$ LOCK-UP AT GRID
$ POINTS 11 AND 12
C5,Q5 .GT. 0.0$ DEFINES C5 IN TERMS OF Q5
DEFINE,@7=3.14159265359$ PI DEFINED
RESET,Q5=ABS(ROT(11)-ROT(12))-@7$ DEFINES Q5
```

In this example, grid points 11 and 12 constitute a lockable joint which locks-up when the fifth logical condition, C5, is true. C5 is true when Q5 exceeds zero; that is, when the magnitude of the relative rotation between grid points 11 and 12 exceeds π. (Note that a DEFINE command has been used to define the value of π.) When C5 becomes true the LOCKUP command is activated and in this example, a linear rotational spring and viscous damper are automatically placed between grid points 11 and 12. The spring and damper are in addition to any that might be defined with the SPRING or DAMPER commands. From the LOCKUP command data, the spring stiffness is 100 and the damper coefficient is 10. When the relative rotational velocity at grid points 11 and 12 becomes equal, the spring and damper will be automatically removed and replaced with a rigid multi-degree-of-freedom constraint which will make the joint rigid and no further relative rotation between the two nodes of the joint will occur.

NOTE 5.17-1: Momentum is always conserved during lock-up, but energy is not generally conserved. The following rules govern the use of the LOCKUP command as they relate to energy conservation.

If only flexible members (MTYPE=0) come into the joint and the user desires that energy be conserved during lock-up, then the spring and damper properties of the LOCKUP command are set to zero.

If any member involved in the lock-up of a joint is rigid (MTYPE=3) then energy cannot be conserved during lock-up and nonzero spring and/or damper properties should be used to remove energy.

Default values for spring and damper properties are,

s1=1000
s2=0
c1=100
c2=0

NOTE 5.17-2: At the present time the LOCKUP command is not available.

5.17.3 - ROTLOCK Command: To Lock-up Rotating Members Using Conservation of Momentum, But Without Detail Analysis in the Neighborhood of the Locking Joint

The ROTLOCK command takes the form,

ROTLOCK:igrid1, igrid2 ? Cj

where,

igrid1 and igrid2 are the grid point numbers involved in the lockup which prior to lock-up have no rotational constraint defined between them

Cj is the j^{th} condition label; a condition label is mandatory for this command

Section 5.18 - Output Control

5.18.1 - General Discussion

The user is given considerable versatility in the form and quantity of output desired. The following results from the program may be output in printed or plotted form:

displacements, velocities, accelerations

internal beam member forces; i.e., axial forces, shear forces, bending moments at both ends of the member.

reactions

mass matrix

tangent stiffness matrix (if STIFF command has been used)

frequencies and mode shapes (if FREQ command has been used)

user defined variables (Q-variables)

status of condition labels

The user may print displacements, velocities and accelerations or internal member forces at selected time intervals or plot displacements, velocities, accelerations or internal member forces in the time or frequency domain. The user may also choose to print the mass or tangent stiffness matrix or to plot a trace (non-zero locations of terms) of these matrices at selected time intervals. Frequencies and mode shapes which have been selected for calculation in the FREQ command are automatically printed. The user may plot the translating, rotating and deforming total geometry of the structure at selected time intervals. In addition, user defined variables and the status of condition labels are automatically printed and may be plotted if desired.

5.18.2 - Sign Conventions For Interpretation of Grid Point Motions and Internal Member Forces

As shown in figure 5.6-1, translation motions, U_X and U_Y are positive parallel to the global X and Y axes respectively. Rotational grid point motion is positive if rotation is in a direction from the positive global X axis to the positive global Y axis.

Also, as shown in figure 5.18-2, member internal axial forces, P_A and P_B are positive parallel to the local V vector. The local V vector runs from end A to end B of the member. (MEMBER commands determine ends A and B. Members run from end A to end B.) Shear forces, S_A and S_B are positive parallel to the local transverse Y' axis and bending moments M_A and M_B are positive in the positive rotational direction.

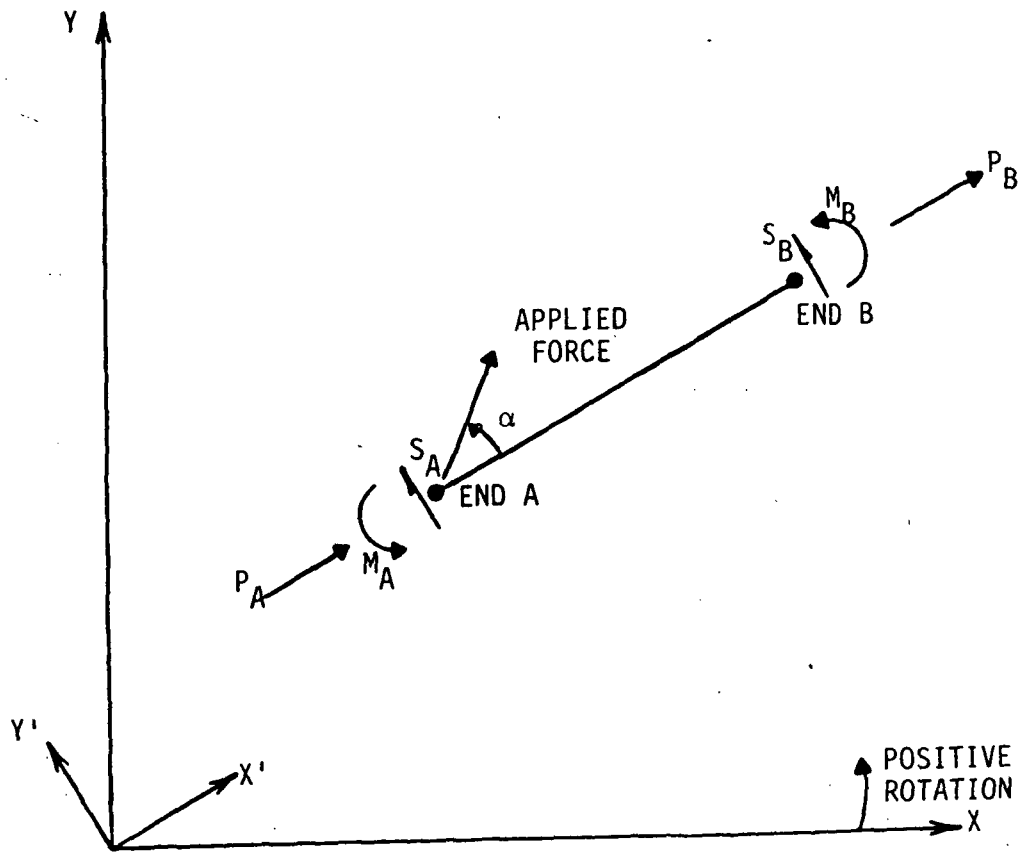


Figure 5.18-1. Sign Conventions For Internal Member Forces

Section 5.18.3: Printing

Printing is accomplished by using the optional PRINT command in conjunction with the GRDPRT and/or MEMPRT commands. Data for the PRINT command takes the form,

PRINT:k1,i1,k2,LG,k3,k4 ? Cj

where,

k1 is an integer specifying the printing of displacements, velocities and accelerations every k1 time steps; if k1 is zero or left blank, there is no request for printing of displacements, velocities and accelerations

i1 is an integer specifying that printing of displacements, velocities and accelerations is to be in the local axis system which rotates with member i1; if i1 is zero or left blank, the global system is used

k2 is an integer specifying the printing of internal forces (axial force, shear force and bending moment at each end of the member) every k2 time steps; if k2 is zero or left blank, no request is made for printing of internal forces.

LG is a letter (either L for Local or G for Global) specifying that printing of internal member forces are to be in the local or global axis system which rotates with each member; if LG is left blank, the Global system is used

k3 is an integer whose absolute value specifies the printing of the mass matrix every |k3| time steps; if k3 is zero or left blank, no request is made for printing the mass matrix; if k3 is positive, the mass matrix is printed; and if k3 is negative traces of the mass matrix are printed

k4 is an integer whose absolute value specifies the printing of the stiffness matrix (including incremental stiffness if requested in STIFF command) every |k4| time steps; if k4 is zero or left blank, no request is made for printing the stiffness matrix; if k4 is positive the stiffness matrix is printed; and if k4 is negative traces of the stiffness martrix are printed

Cj is the j th optional condition label defined with a Cj command; see section 5.15 for further details

NOTE 5.18-1: In addition to the even increment printing specified in the PRINT command, printing will always occur at the start time in the TIMSPAN command and when certain condition statements become TRUE.

Example 5.18-1

```
TIMSPAN:10.,2000.
PRINT:100,5,100,5,100,0
```

In this example, the mass matrix, displacements, velocities, accelerations and all internal member forces (axial forces, shear forces and bending moments at both ends of designated members) are to be printed at the starting time of 10 as given in the TIMSPAN command and every 100 time steps. Results will be given in the time varying local coordinate system which rotates with member number 5. The tangent stiffness matrix is not printed. Note that the first entry following the command name instructs printing of displacements, velocities and accelerations simultaneously at designated grid points, while the fourth entry following the command name instructs printing of all internal forces simultaneously in the designated members. Designated grid point numbers and member numbers for printing are provided using the commands GRDPRT and MEMPRT described next.

Unless otherwise specified, printed output of displacement, velocity and accelerations will occur for all grid points and printed output of forces will occur for all members. The commands GRDPRT and MEMPRT limit printed output to subsets of grid points and members.

The GRDPRT command takes the form,

GRDPRT,igrid1,igrid2,...,

where,

igrid1, igrid2,..., constitutes a list of grid points by specifying actual grid point numbers and/or predefined list(s) of grid point numbers (see page 5.2-2)

Example 5.18-2: Use of GRDPRT Command

```
LIST,10,1,2,3,5T15B5 $ DEFINE A LIST OF
$ FOR LATER USE
GRDPRT,L10,4 $ SUBSET OF GRID POINTS
$ FOR PRINTING
```

In this example, a list of grid points was defined somewhere in the data set. This list was denoted as "10" and contains grid points 1, 2, 3, and 5, 10 and 15. The GRDPRT command creates a subset of grid points for printing by referencing the previously defined list as "L10" and adding to it grid point 4.

The MEMPRT command takes the form,

```
MEMPRT,imem1,imem2,...,
```

where,

imem1, imem2,..., constitutes a list of member numbers by specifying actual member numbers and/or predefined list(s) of member numbers (see page 5.2-2)

Example 5.18-3: Use of MEMPRT command

```
LIST,8,20T40B5 $ DEFINE A LIST OF
$ FOR LATER USE
MEMPRT,L8 $ SUBSET OF MEMBER NUMBERS
$ FOR PRINTING
```

In this example, a list of member numbers was defined somewhere in the data set. This list was denoted as "8" and contains 20, 25, 30, 35 and 40. The MEMPRT command creates a subset of member numbers for printing by referencing the previously defined list as "L8".

Reactions are printed by using the REACT command. Data takes the form,

REACT,igrid,idir,m

where,

igrid is a grid point number

idir is a direction; idir=1 for x-direction, idir=2 for y-direction and idir=3 for rotation

m is the member number of the local coordinate system in which the reaction is to be calculated; m equal to zero or left blank indicates the global axis system

NOTE 5.18-2: Reactions associated with grid points and load directions specified in a REACT command which are not currently dependent degrees-of-freedom due to an active SDFC or MDFC command, or rigid member constraint, or for some other reason, are calculated as zero. See also Notes 5.13-1 through 5.13-3.

A condition label cannot be used in conjunction with a REACT command.

5.18.4: Plotting

If results are to be plotted, the program must be instructed to create an plot output file. Any or all of the data on the plot output file can be accessed through the use of plotting commands. The plotting commands provide instructions to the post-processor and may either be placed in the data set and thereby executed immediately following the completion of all LATDYN calculations or used at a later time during a post-processing session. Thus any data that has been placed on the plot output file and appropriately saved on disk or magnetic tape, can later be accessed and the results plotted. (For further discussion see post-processing in Appendix A.)

5.18.4.1: Creating a plot output file

Two commands are required to instruct creation of the plot output file. They are,

PLOTINC - specifies the density of data points used to generate plots

PLOTLIMIT - specifies the time duration during which data is placed on the plot output file

These commands take the form,

PLOTINC,r?Cj

where,

r is an integer specifying that all plots are to be generated from data calculated every r th time step

Cj is the j th optional condition label; see section 5.15 for further details

NOTE 5.18-3: Multiple PLOTINC commands may be used, but they have a condition-order priority as discussed in NOTE 5.15-1 of Section 5.15. Thus, the first appearance in the data set of a PLOTINC command cannot have an attached condition label. See also Section 4.5.

PLOTLIMIT,t1,t2

where,

t1 is the lower time limit for all data on the plot output file

t2 is the upper time limit for all data on the plot output file

NOTE 5.18-4: If a PLOTLIMIT command is not used, the time limits of the TIMSPAN command are employed.

5.18.4.2: Post-processor commands

Once the LATDYN execution of results is complete, the post-processor can be used to plot any quantities placed on the plot output file. The user instructs the post-processor using commands which fall into one of six categories,

(1) generation of time history plots
(2) generation of user defined variables (Q-variables) plotted aganist one another
(3) generation of pictoral view of structure during motion and deformation
(4) generation of frequency domain plots
(5) zooming for macro and micro viewing of time and frequency domain plots
(6) user assistance during post-processor interactive session

The post-processor commands may appear in the data set and be executed directly after LATDYN computations are finished, or may be used at a later time in an interactive session with the post-processor.

NOTE 5.18-5: The use of a command in category (1) generates a set of time histories of response on one plot; that is, the time variation of all designated quantities in the command will appear on one set of axes.

Post-processing Commands - Category 1

The following commands generate time history plots of data that reside on the output file.

PLOTXLOC,n1,n2,n3,...	to plot time histories of global X position at prescribed grid points
PLOTYLOC,n1,n2,n3,...	to plot time histories of global Y position at prescribed grid points
PLOTROT,m1,m2,m3,...	to plot time histories of angular position of prescribed members; positive angle measures in direction from positive global x to positive global Y axis

where,

n1,n2,n3,... represents a set of grid points numbers

m1,m2,m3,... represents a set of member numbers

Post-processing Commands - Category 1 (continued)

The following commands generate time history plots of data that resides on the output file.

PLOTMAG,p,n1,n2,n3,....	to plot time histories of magnitude of motion quantities (displacement, p=D; velocity, p=V; or acceleration, p=A) at a set of grid points
PLOTX,p,n1,n2,n3,...	to plot time histories of motion quantities (displacement, p=D; velocity, p=V; or acceleration, p=A) parallel to global X axis at a set of grid points
PLOTY,p,n1,n2,n3,...	to plot time histories of motion quantities (displacement, p=D; velocity, p=V; or acceleration, p=A) parallel to global Y axis at a set of grid points
PLOTPHI,p,n1,n2,n3,...	to plot time histories of rotational quantities (displacement, p=D; velocity, p=V; or acceleration, p=A) at a set of grid points; positive angular rotation is in a direction from from positive global X axis to positive global Y axis

where,

p is a parameter designated as ,

D - for displacement
V - for velocity
A - for acceleration

l.....represents the member number of a local coordinate system; if zero or blank the global axis system is assumed

n1,n2,n3,... represents a set of grid points numbers

The following commands generate time history plots of data that resides on the output file.

PLOTXP,l,p,n1,n2,n3,... to plot time histories of motion quantities (displacement, p=D; velocity, p=V; or acceleration, p=A) parallel to a local X' axis of member l at a set of grid points

PLOTYP,l,p,n1,n2,n3,... to plot time histories of motion quantities (displacement, p=D; velocity, p=V; or acceleration, p=A) parallel to a local Y' axis of member l at a set of grid points

PLOTANGLE,p,n1,n2,n3,...to plot time histories of angular orientation of a motion quantities (displacement, p=D; velocity, p=V; or acceleration, p=A) at a set of grid points; positive angular orientation is measured in a direction from positive global x axis to positive global y axis

PLOTSTRAIN,m1,m2,m3,... to plot time histories of axial strain in a set of members

PLOTFEXT,ma,ld1,ld2, to plot external force magnitude if ma is M or orientation angle if ma is A for APPLOAD sets designated by load1, load2,...

where,

p is a parameter designated as ,

D - for displacement
V - for velocity
A - for acceleration

n1,n2,n3,... represents a set of grid points numbers

m1,m2,m3,... represents a set of member numbers

l.....represents the member number of a local coordinate system; if zero or blank the global axis system is assumed

Post-processing Commands - Category 1 (continued)

The following commands generate time history plots of data that reside on the output file.

PLOTC,c1,c2,c3,...	to plot TRUE or FALSE time histories of a set of condition labels
PLOTQ,q1,q2,q3,...	to plot time histories of a set of user defined variables
PLOTAX,m1,m2,m3,...	to plot time histories of axial forces in a set of members (tension is positive)
PLOTSHA,m1,m2,m3,...	to plot time histories of shear force at end A in a set of members
PLOTSHB,m1,m2,m3,...	to plot time histories of shear force at end B in a set of members
PLOTMOMA,m1,m2,m3,...	to plot time histories of bending moments at end A in a set of members
PLOTMOMB,m1,m2,m3,...	to plot time histories of bending moments at end B in a set of members

where,

p is a parameter designated as ,

D - for displacement
V - for velocity
A - for acceleration

m1,m2,m3,... represents a set of member numbers

c1,c2,c3,... represents a list of condition label numbers

q1,q2,q3,... represents a list of user defined variable numbers

Post-processing Commands - Category 2

The following command creates a plot of user defined variables (Q-variables) where one user defined variable is plotted aganist another. Data takes the form,

PLOTQVQ,Q1,Q2,Q3,...

where user defined variables Q2, Q3, etc. are plotted aganist Q1.

Post-processing Commands - Category 3

The following command generates pictoral views of the structure

SNAPSHOT,ts,te,k,F,M,N to plot the displacement/deformation of the structure every k th time step from time "ts" to "te" with drawn forces if F appears; member numbers if M appears; grid point numbers if N appears; drawn axes if A appears; and overlaid snapshots if O appears

When the SNAPSHOT command is used without the presence of ts and te, displacement/deformation plots of the structure are generated at the following times:

starting time as prescribed in the TIMSPAN command, (thus producing an initial deformation plot),

at time t1 as prescribed in the PLOTLIMIT command

and every k th time step after time t1.

Post-processing Commands - Category 4

Results on the output file may be reduced to produce frequency domain plots. Power spectral density and power spectral level plots may be generated. Data takes the form,

PSD - for power spectral density (units are power/Hertz)

PSL - for power spectral level (units are power/spectral resolution bandwidth)

The PSD and PSL commands apply to the time history command which immediately proceeded them.

Post-processing Commands - Category 5

The TIMEWINDOW command is used to set the axes of the time history plots. Effectively the user can zoom-in on results of interest.

The command takes the form,

TIMEWINDOW,TIME1,TIME2

where TIME1 and TIME2 provide upper and lower bounds for the time scale on time history plots and plots of Q-variables versus one another. The time window must be within the bounds of the PLOTLIMIT command. If TIME1 and TIME2 are left blank, the bounds of the PLOTLIMIT command are used.

The position in the data sequence determines which time history plots are affected by a particular TIMEWINDOW command as each command remains active until another appears in the data set.

The FREQWINDOW command is used to set the axes for plots in the frequency domain. Effectively the user can zoom-in on results of interest.

The command takes the form,

FREQWINDOW,FREQ1,FREQ2

where FREQ1 and FREQ2 provide upper and lower bounds for the frequency scale on plots in the frequency domain.

NOTE 5.18-6: The position in the data sequence determines which frequency domain plots are affected by a particular FREQWINDOW command as each command remains active until another appears in the data set. PSD and PSL commands must follow a FREQWINDOW command.

Post-processing Commands - Category 6

During an interactive post-processing session the user can get assistance from the post-processor program via the commands HELP and INFO.

The HELP command provides on-line quick reference to a summary of post-processor commands and their function.

The INFO command provides on-line quick reference to important characteristics of the data on the output file and current status of the post-processor. This includes the job title and subtitle, time limits, current time window, number of time steps, grid point and member numbers, Q-variable identification numbers and condition label numbers. The user will find this command very useful, since what is on a particular data file may be forgotten.

5.18.4.3 Plotting Examples

Example 5.18-3: Plot Displacements in Global x-Direction

```
PLOTINC,100$ SET PLOTTING DENSITY
PLOTLIMIT,0.0,20.0$ SET UPPER AND LOWER TIME
$ LIMITS FOR PLOTS
PLOTX,D,2,4,6,8$ PLOT DISPLACEMENTS AT GRID
$PTS. 2;4,6,8
```

In this example, the plotting specifications, as supplied via the PLOTINC and PLOTLIMIT commands, are the generation of time history plots from time 0 to 20 using results calculated every 100 time steps. The PLOTX command indicates that quantities to be plotted are components of vectors parallel to the global x axis. The plot parameter "p" being set to D and the string of integers indicate that displacement time histories are desired at grid points 2, 4, 6 and 8. Since only one PLOTX command is used, a single plot is created containing the four time histores of the four designated grid points. Scaling of the plot will be performed automatically on the basis of the results for all four histories.

Example 5.18-4: Plot Acceleration in Local System

```
PLOTINC,100$ SET PLOTTING DENSITY
PLOTLIMIT,0.0,20.0$ SET UPPER AND LOWER TIME
$LIMITS FOR PLOTS
PLOTYP,2,A,2,4$ ACCEL PLOTS TRANVERSE TO
$MEMBER 2 AT     $ POINTS 2,4
```

In this example, the PLOTYP command indicates quantities to be plotted are components of vectors parallel to a local y' axis which is transverse to a structural member. The first integer value indicates that the local axis is associated with structural member 2. The parameter "p" being set to "A" and the string of integers which follow, indicate that time histories of acceleration are desired at grid points 2 and 4. Plot specifications are the same as in the previous example.

Example 5.18-5: Plot Bending Moment

```
PLOTINC,100$ SET PLOTTING DENSITY
PLOTLIMIT,0.0,20.0$ SET UPPER AND LOWER TIME
LIMITS FOR $ PLOTS
PLOTMOM1,12,20$ INTERNAL BENDING MOMENT
```

In this example, the PLOTMOM1 command indicates quantites to be plotted are bending moments at end 1 of designated members. The designated members are prescribed by the string of values, namely, 12 and 20. The plot specifications are the same as those of Example 6.18-3.

Example 5.18-6: Use of Condition label to output additional plots during certain phases of the calculations

```
PLOTINC,100$ SET PLOTTING DENSITY
PLOTINC,25? C10$ SET ALTERNATE PLOTTING
$DENSITY
C10,& $ CONTINUATION LINE FOLLOWS
(ABS(SH1(12)) .GT. 1.E4 .OR. ABS(SH1(20))&
.GT. 1.E4)  $ PREVIOUS LINE SETS CONDITION
$LABEL C10
PLOTLIMIT,0.0,20.0$ SET UPPER AND LOWER TIME
$LIMITS FOR PLOTS
PLOTSHA,12,20$ SHEAR FORCES IN MEMBERS 12 AND
$20
```

In this example, the PLOTSHA command indicates that time histories of shear forces at end 1 of designated members are to be plotted. Designated members are 12 and 20. Plotting normally uses results generated every 100 time steps, but when condition C10 is TRUE, (magnitude of shear force in either member exceeds 10,000 units), plotting uses results generated every 25 time steps.

Example 5.18-7: Plot Magnitude of Velocity Vector

```
PLOTINC,100$ SET PLOTTING DENSITY
PLOTLIMIT,0.0,20.0$ SET UPPER AND LOWER TIME
$LIMITS FOR PLOTS
PLOTMAG,V,5,10,15,20$ MAGNITUDE OF VELOCITY
$VECTOR AT GRID POINTS 5, 10, 15 AND 20
```

In this example, the PLOTMAG command indicates that time histories associated with the magnitude of a designated quantity is plotted. The "V" entry indicates the designated quantity is velocity. The following string of integers are the grid points at which the magnitudes of the velocity vectors are to be plotted.

Example 5.18-8: Undeformed and Deformed Structural Plots

```
TIMSPAN,0.0,50.0$ STARTING AND STOPPING TIME
PLOTLIMIT,6.0,15.0$ PLOT FROM TIME OF 6 TO 15
SNAPSHOT,,1000$ DEFORMATION PLOTS EVERY 1000
$TIME STEPS
```

In this example, structural deformation plots are indicated at time equal to 0, (the initial time in the TIMSPAN command), time equal to 6, (the initial plot time in the PLOTLIMIT command), and every 1000 time steps thereafter up to time equal to 15, (the final time in the PLOTLIMIT command).

SECTION 6 - USER GUIDELINES

User guidelines for selection of the following data values are provided in this section:

Time Step In TIMSTEP Command
Increment For Mass Matrix Update In INCMASS Command

Section 6.1 - Time Step In TIMSTEP Command

Selection of the time step is very important to successful program execution. When using any explicit time integration operator (INTGTYP set to 1 through 4), the time step must be sufficiently small so as to avoid numerical instability. In linear problems it can be shown that the time step must be less than twice the reciprocal of the highest system frequency. However, most of the problems that LATDYN would normally be used for are nonlinear. Nevertheless, the linear criterion for numerical stability usually provides a good guideline when the highest instantaneous system frequency is used in the criterion. This frequency can be derived from the current tangent stiffness and mass matrices. However, this information is not generally available to the user prior to program execution. It is suggested then that the following procedure be used when a satisfactory selection of the initial time step cannot be made prior to program execution.

The suggested procedure for determining a satisfactory initial time step is a trial and error approach based on the general behavior of explicit time integration algorithms. In general, for these algorithms, the largest time step for which the solution is numerically stable is also the most efficient and provides results which are nearly as accurate as any smaller time step. Thus, the procedure is to start with a numerically stable time step and increase that time step until numerical instability is found and then, to the degree desired, iterate to the largest stable time step. The suggested steps in this procedure follow.

(1) Find the highest system frequency which usually will be associated with a localized vibration mode. To do this, locate the shortest flexible finite element of the model. Calculate its bending and extensional vibration frequencies using the formulas,

$\omega = (kL)^2 c/sL$ and $\omega = c/(2L)$

and select the higher value where,

kL is the characteristic beam value and it is suggested to use the value for a fully clamped beam, namely, 4.730

c is the elastic wave speed in the member; that is, $\sqrt{(E/\rho)}$ which is about 200,000 inches/second for aluminum

L is the element length

s is the slenderness ratio; that is, $L/[\sqrt{(I/A)}]$

E is Young's modulus

ρ is the mass density per unit volume

I is the element cross sectional moment of inertia

A is the element cross sectional area

(2) Perform a series of small trial runs to establish a more efficient time step. This may be done by executing the program, (preferably interactively), using $2/\omega_{max}$ as a first choice for the time step and using a terminating time in the TIMSPAN command equivalent to about 100 time steps. If results grow unbounded, reduce the time step. When a stable time step has been found, double its value and rerun. Continue this process until results grow unbounded.

(3) Use the last stable time step or if desired iterate to find a better selection. However, it is not usually advantageous to expend much effort in iteration since the problem being solved is usually nonlinear and could possibly become unstable later in the analysis.

(4) Execute program with selected time step. If INTGTYP is 3 or 4, time step will automatically be adjusted and numerical solution will likely remain stable. In any case, the time step may be changed during execution via user options in the input data. The following partial runstream serves as an example in accomplishing this:

Example 6-1: Runstream For User Control of Time Step Based on Program Calculation of Highest System Frequency

```
STIFF,1000,N$ UPDATE TANGENT STIFFNESS MATRIX
INCMASS,1000$ UPDATE MASS MATRIX
FREQ,1000,1,0$ CALCULATION AND PRINTOUT OF HIGHEST
$FREQUENCY
RESET,Q1=0.95*(2/OMEGA)
TIMSTEP,(selected value from steps 1 to 3)$ INITIAL TIME
$ STEP
C5, T .GE. 1000*(selected time step from steps 1 to 3)
TIMSTEP,Q1?C5$ NEW TIME STEP EQUALS Q1
```

In this sample partial runstream, the highest system frequency is calculated by the program every 1000 time steps using updated stiffness and mass matrices. A user defined variable, in this case Q1, is defined as twice the reciprocal of the highest frequency and an additional 5% reduction is arbitrarily added. The variable Q1 is used as the new time step after the first 1000 time steps. A condition label, in this case C5, is defined to activate the new time step after the initial time step has been used 1000 times.

NOTE 6.1-1: It is not advisable to calculate the highest frequency too often during a run as this can increase computer solution time considerably.

Selection of member type, MTYPE in the MATPROP command, influences time step selection. If a member is very stiff and is treated as a flexible member, MTYPE=0 in the MATPROP command, with a high stiffness or cross sectional property, the highest frequency of the system will be greatly increased, thus decreasing the necessary time step for numerical stability when INTGTYP is 1 through 4. It would then be desirable to use a rigid member, MTYPE=3 in the MATPROP command, as this would eliminate the high frequency from the problem. However, the use of rigid members requires updating of the mass matrix as discussed in the next section. Consequently, the user must trade-off the small time step of stiff members against the mass matrix updating. Each results in incresed computer solution time. In general, though, reduced time step has a greater adverse affect on solution time, than mass matrix updating.

Section 6.2 - Incremental Updating of Mass Matrix

LATDYN uses a consistent mass formulation in its solution procedure. Because of the way in which the finite elements are modeled, the consistent mass matrix will change very little during program execution even if large angular changes between members occur, provided all members are fully flexible, MTYPE=0 in the MATPROP command. Thus when all members are flexible, few if any updates to the mass matrix are required for large angular motions of members. Other considerations, such as updating of the mass matrix for frequency calculation as used in Section 5.1 will then govern the use of INCMASS.

If rigid members, MTYPE=3 in the MATPROP command, are used, the mass matrix will need to be updated. The updating increment depends upon the angular rotation rate of the rigid members. That is why the INCMASS command permits an incremental angle entry which instructs mass matrix updates when any rigid member rotates through the specified incremental angle since the previous update, independent of what caused the previous update. The default value of five degrees for this angle is based on studies performed on mass matrix updating. The sample problem of Section 6.1 illustrates the effect of these updates on solution accuracy. As described in Section 5.3, mass matrix updating will also occur at even increments of time steps in accordance with the first numerical entry in the INCMASS command.

Mass matrix updating must also occur when constraints become activated or deactivated. This is handled automatically by the program. The user has no control over such updates; they are in addition to to the periodic update specified in the INCMASS command.

Independent of what caused a mass matrix update, a message about its occurrence will appear in the output file for user information.

SECTION 7 - SAMPLE INPUT AND OUTPUT

Presented in this section of the User's Manual are four representative sample problems which illustrate the use of LATDYN. Included with each problem are the actual input data cases along with a brief description of the commands used. In addition, sample output is presented in the form of printed output from LATDYN and various plots generated through the postprocessor.

Problem 7.1 - Rotating Rigid Beam

This sample problem consists of a single rigid beam having concentrated tip masses and given an initial rotational velocity as shown in Figure 7.1-1. The main purpose of this example is to illustrate the use of all the required commands, some of the optional commands, comments, continuation lines and the free field input data format.

Input Data

The input data for this problem is shown in Figure 7.1-2. This case uses in addition to the required commands, 6 optional commands, each of which is discussed below (Note that although the SUBTITLE command is optional, it is considered essential and discussed along with the other required commands).

Listed below are the required commands necessary for successful execution of LATDYN,

TITLE
SUBTITLE (See note above)
PORDER
TIMSPAN
TIMSTEP
MATPROP
GRID
MEMBER

Note that although the program will execute without the PRINT or PLOTINC commands, it is essential to use at least one of them in order to get any meaningful output from the program.

The TITLE and SUBTITLE commands each consist of one line of alphanumeric data which is printed out as part of the output. The user is encouraged to keep accurate descriptions through these commands for ease in identifying individual data cases.

The PORDER command specifies the maximum number of finite elements and grid points in the model. To maximize program efficiency it is recommended that the values specified be the same as the actual values in the problem. In this example, one beam element and two nodes are specified.

The TIMSPAN command gives the starting and terminating times for the analysis. For this problem, the start and stop times are 0.0 and 1.0 respectively. Note that through the use of a STOP command (used in Problem 7.2) the user can terminate the analysis prior to the specified terminating time if a previously specified condition becomes true.

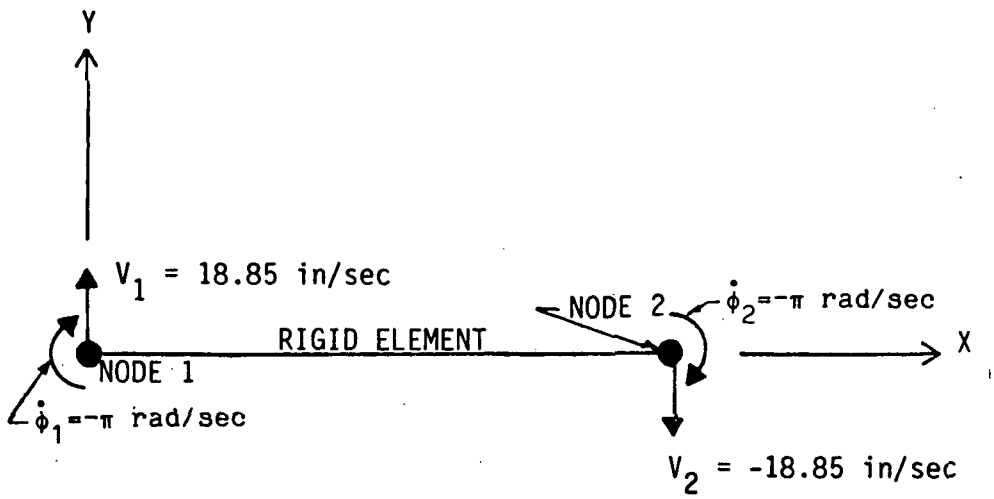


Figure 7.1-1 Rotating Rigid Beam With Initial Velocities

```
$
$THIS SAMPLE PROBLEM IS A RIGID BEAM WITH ATTACHED TIP MASSES WHICH
$HAS AN INITIAL ROTATIONAL VELOCITY SUCH THAT THE BEAM WILL ROTATE
$EXACTLY PI RADIANS IN ONE SECOND.
$
$TITLE: DATA CASE TITLE (UP TO 80 CHARACTERS)
TITLE: T9S - FIRST TEST CASE FOR LATDYN USER'S GUIDE
$
$SUBTITLE:DATA CASE SUBTITLE(UP TO 80 CHARACTERS)
SUBTITLE: ONE RIGID ELEMENT WITH INITIAL VELOCITY
$
$DEFINE: @N=CONSTANT
DEFINE: @1=.0002588      $DEFINE MATERIAL DENSITY
DEFINE: @2=3.141592654    $ DEFINE PI
        @3= 1.0E-3  $DEFINE VALUE OF CONCETTRATED MASS
$
$PORDER: NELM,NODES
PORDER:  1,   2       $ONE BEAM ELEMENT AND TWO NODES ARE DEFINED
$
$TIMSPAN: START,STOP
TIMSPAN: 0.0,   1.0  $ANALYSIS TO RUN FOR 1 SECOND
$
$TIMSTEP:DELTA T ?CONDITION LABEL
TIMSTEP: 1.0E-3       $INTEGRATION TIME STEP IS .001 SECONDS
$
$MATPROP:MATNAME,MTYPE, E   ,  A    ,   I   ,  RHO ? CONDITION LABEL
MATPROP: ALUMINUM, 3, 10.5E06, @2, .78540, @1   $ USING RIGID MEMBER
$ WITH ALUMINUM MATERIAL PROPERTIES
$
$GRID:NST,# NODES, X1  , Y1    ,X2    , Y2   ,INC
GRID:  1,    2,    0.0,  0.0,  12.0, &
  0.0,  1  $THIS IS A CONTINUATION OF THE LINE ABOVE -- TWO GRID POINTS
$ HAVE BEEN DEFINED
$
$MEMBER:IMEMB ,N1   ,N2   ,NE    ,MAT
MEMBER: ,1,   2,    1,    ALUMINUM    $ONE MEMBER FROM NODES 1 TO 2 USING
$THE PROPERTIES OF "ALUMINUM" FROM THE MATPROP COMMAND
$
$
$PRINT:DVA INC, COORD,INTF INC,COORD,MM INC,SM INC ?CONDITION LABEL
PRINT:   100,    0,     100,     L,     0,    0  $PRINT DISPLACEMENTS/
$VELOCITIES/ACCELERATIONS EVERY 100 TIME STEPS AND FORCES EVERY 100
$
$PLOTINC: PLOT INC  ? CONDITION LABEL
PLOTINC:    10   $STORE RESULTS EVERY 10 TIME STEPS FOR PLOTTING
$
$INCMASS: NSTEPS,ANGLE?CONDITION LABEL
INCMASS:     1, .0875       $UPDATE MASS MATRIX EACH TIME FOR RIGID ELEMENT
$
```

Figure 7.1-2 Input Data For Problem 7.1

```
$ADMASS: NDE1,  IDIR1, NDE2,  IDIR2, MASS   ?CONDITION LABEL
ADMASS:     1,     1,     1,     1,     @3   $LUMPED MASSES ARE DEFINED AT EACH
            1,     2,     1,     2,     @3   $NODE ACTING IN THE X AND Y DIRECTIONS
            2,     1,     2,     1,     @3   $WITH MAGNITUDE AS GIVEN IN THE DEFINE
            2,     2,     2,     2,     @3   $COMMAND
$
$
$VEL,GRID PT,   DIR, COORD ,INIT VEL
VEL:      1,      3,      0,     -@2      $INITIAL ROTATIONAL AND NORMAL
          2,      3,      0,     -@2      $VELOCITIES ARE DEFINED AT
          1,      2,      0, +18.849556 $GRID POINTS 1 AND 2
          2,      2,      0, -18.849556
--EOI/TOP--
??
```

Figure 7.1-2 continued

The TIMSTEP command indicates the integration step size for the transient response. Note that both the TIMSPAN and TIMSTEP commands must use a consistent set of time units. The specified time step size for this example is .001. The TIMSTEP command can be modified via a condition label to provide variable step sizes during the analysis.

Material and structural properties of the beam members are specified by the MATPROP command. Each beam member has a unique identification name (composed of any alphanumeric characters) which is specified by the MEMBER command to designate the properties of the elements being defined. The Young's modulus (E) and the mass density/unit volume (RHO) are specified material properties whereas the cross sectional area (A) and the area moment of inertia (I) are specified structural properties. Note that all of the above quantities must be specified in a consistent set of units. In this sample problem, the material and structural properties of an aluminum bar with circular cross section and radius of 1 have been specified using the foot/inch/pound/second system of units. A unique feature of LATDYN is the ability to vary these properties during execution of the program by using a condition label associated with the MATPROP command. Further discussion on this feature can be found in Note 5.4-5.

Grid points of the finite element model are defined using the GRID command (Note that in this manual the terms grid point and node are used interchangeably). Using a single GRID command the user can generate a series of equally spaced grid points between two specified points or define a single grid point. In this example, the beginning and ending coordinates of two grid points are specified as X1, Y1, X2, Y2 and the starting grid point number is defined as grid point 1. Then since the number of grid points and the increment in numbering are specified as 2 and 1 respectively, two grid points are defined. Note that although the GRID command can be used to define a string of grid points, the user must use a MEMBER command to ensure that the grid points are connected properly.

Finite element beam members are defined through the MEMBER command. As in the previous command, this command can be used to define a string of members or a single member. The first parameter of the MEMBER command is an optional user defined member number which allows the user to specify the numbering of the elements. If the member number is not specified then the program automatically assigns unique member numbers. In addition, the user provides the beginning and ending grid point numbers of the first member and the number of members being defined with that particular command. If the number of members defined is equal to 1, then a single beam member is defined between the two grid point numbers specified, however, if the number of members is greater than 1, then a string of members is defined as starting from the ending grid point of the first member and continuing until the

total number of members in that string is equal to that specified on the command. The increment between the beginning and ending grid point numbers of each of the remaining members in the string is assumed to be the same as that of the first member. In addition, the properties of each member are labeled with an identification name which must correspond with the identifier of a valid MATPROP command. In this example, one member is defined between grid points 1 and 2 and the properties of the member are given on the MATPROP command with the name "Aluminum".

The optional commands used in this example are,

DEFINE
PRINT
PLOTINC
INCMASS
ADMASS
VEL

Each of these are discussed below.

The DEFINE command is a convenient way for the user to define constants which are used often in the data set. The form of this command is

@n=Constant

where the "n" is a unique integer from 1 to 99 (thus up to 99 constants can be defined). Once a constant is defined, the shorthand definition can be used anywhere in the data case and the preprocessor will substitute the appropriate constant at that point. In this case, three shorthand definitions (@1, @2, and @3) appear in the MATPROP, ADMASS, and VEL commands.

The PRINT command is the means for achieving printed output. The user can request the printing of displacements, velocities, and accelerations as well as internal forces (axial force, shear force, and bending moment) in either a local or global coordinate system. In addition, provisions exist for the printing of the mass and stiffness matrices. With the PRINT command, the user specifies the increment at which each of the above quantities will be output. Thus in this example, the displacements, velocities, and accelerations of all the grid points will be printed every 100 time steps in the global coordinate system, and the internal forces of all the members will be output every 100 time steps in the local coordinate system of each respective member. Also, no request has been made for the periodic printing of the mass and stiffness matrices. The PRINT command can be controlled via a condition label such that the user can vary the frequency and content of the output during different portions of the analysis.

The ability to create plots is obtained by using the PLOTINC command. This command consists of a user specified integer, "r", which indicates that at every "r" time steps all of the results

will be stored on a seperate file which can be accessed through the postprocessor. The user is encouraged to read Section 5.19.4 for the details of the various plotting options available through the postprocessor. An important point to note is that the PRINT and PLOTINC commands are completely independent since they operate on different user specified increments. In this example, the results are sampled every 10 time steps so that the plots of this analysis will contain 10 times more data than that printed. The plotting increment on the PLOTINC command can also be changed by a condition label if desired. If the user saves the plot files as described in Section 5.19 and the Appendices, the postprocessor can be executed separately from LATDYN after the LATDYN execution is complete.

Periodic updating of the system mass matrix is accomplished via the INCMASS command. The user specifies an integer "n" and the mass matrix is updated every "n" time steps (Note that the mass matrix will not be printed out unless the user specifies the same increment on the PRINT command). In addition, the user can specify the maximum angular rotation that any rigid member can undergo without the mass matrix being updated. The default value of this rotation is .0875 radians (5 degrees). See Section 6.2 for a discussion of the necessity of updating the mass matrix and its affect on system momentum. For this problem, the mass matrix is updated at every time step and the value of the rotation angle is specified as .0875. INCMASS can also be altered during execution by a condition label.

Inertial coupling of degrees-of-freedom or lumped masses can be added to the model via the ADMASS command. In this example, lumped masses have been added to grid points 1 and 2 such that the masses affect the x and y directions but not the rotational direction. Note that the @3 which was defined by an earlier DEFINE command is used to specify the magnitude of the lumped mass. It is possible, through the use of a condition label, to add or remove the ADMASS command or to change the data on an existing command. See Section 6.2 for a discussion of the affect on system momentum due to mass additions during execution.

Initial velocity conditions are imposed on the structure through the use of a VEL command. Velocities can be prescribed in the x, y or rotational directions either in the global coordinate system or in the local system of a designated member. In this case, both rotational and normal initial velocities have been prescribed for both grid points such that the beam will rotate exactly π radians in one second.

Several points can be made from an examination of the data in Figure 7.1-2. First, note the use of comments (which are preceded by a "$" sign) can be used anywhere in the data. Second, the free field format allows any number of spaces between data items. Thus, the number 1.0 E-3 would be interpreted as two pieces of data and should be input as 1.0E-3. Third, if a command is used

several times in succession, it is not necessary to repeat the command name since the program will assume all lines of data apply to that name until it encounters a new name (as demonstrated in the DEFINE, ADMASS and VEL commands of the sample problem). Fourth, a line of data can be continued to the next line by using a "&" and completing the remaining data on the following line (as demonstrated with the GRID command).

Sample Printed Output

Shown in Figures 7.1-3a through 7.1-3f is the printed output resulting from the execution of LATDYN using the data set of Figure 7.1-2. Figures 7.1-3b through 7.1-3d show how the input data is printed out after it has been read and assimilated by LATDYN (Note that the order that the commands are output is not necessarily the same as they were input). It is often useful to compare this information with the data case to ensure that all of the data provided by the user is correct for the purpose it was intended.

Following the input data printout, the program outputs any preliminary computations and presets which were carried out (Figure 7.1-3d). First the mass properties of the structure are printed. These entail the location of the centroid, the total mass, and the mass moments of inertia about the x, y and z axes. Next, a summary of the constraints imposed on the structure is listed. For this problem, the maximum number of constraints is given as three since for a two noded rigid beam element only three of the six degrees-of-freedom are independent. Then any preset displacements and velocities imposed by a DIS or VEL command are displayed. Also, information concerning the dynamic direct access memory usage is output.

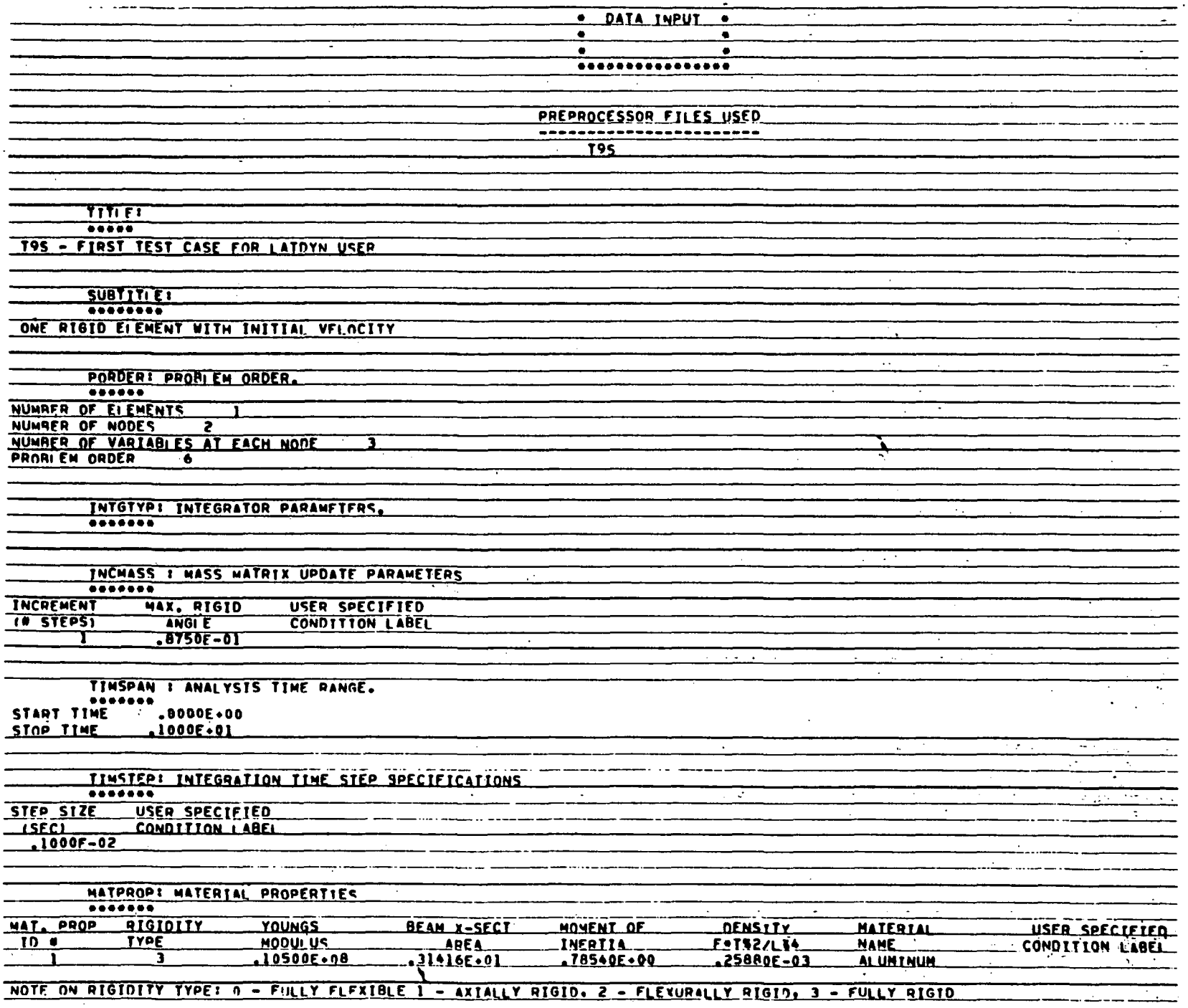
Next, as also shown in Figure 7.1-3d, an informative message is printed dealing with the integration procedure and the manner in which the displacements, velocities and accelerations are computed at each time step. It should be noted that most of the printed output mentioned above is typical of that which each run of LATDYN will generate, however, the printing of the assimilated input data and some of the preliminary computations and presets will vary from case to case depending on the input.

Given in Figures 7.1-3e and 7.1-3f are the results for this case at selected time increments. For every case in which a PRINT command is used, the output will always begin with the information known at time zero before the first integration step is taken. Then results will be printed according to the increment which the user specified on the PRINT command. In this example, the displacements, velocities and accelerations as well as the forces were specified to be output at every 100 time steps, thus with a time step of 0.001 the next printout is at 0.100 seconds (Figure 7.1-3e). For the case of a rigid beam, the values of the internal forces and moments are zero. The first two columns of results

listed under the heading "SOLUTION VECTORS" consist of the sequence number (the order in which the grid points were entered in the data) and the grid point number (the actual number which the user wishes assigned to the grid point). Listed under the subheading "GRID POINT TRANSLATION" and "GRID POINT ROTATION" are the x, y and phi displacements, velocities and accelerations of each of the grid points in its own local coordinate system. Directly underneath the local coordinate information are the values of "XLOC" and "YLOC" which are the coordinates of each node in the global system. Finally, the results listed under the heading "MEMBER DATA" are the rotations and strains of each beam member.

Figure 7.1-3f shows the printout given for the last time step in the analysis which also includes a summary of the cpu time used in the major subroutines and assembling/inverting the mass matrix.

Figure 7.1-3a



```
                    • DATA INPUT •
                    •            •
                    •            •
                    ••••••••••••••

                    PREPROCESSOR FILES USED
                    -----------------------
                    T9S

        TITLE:
        *****
T9S - FIRST TEST CASE FOR LATDYN USER

        SUBTITLE:
        ********
ONE RIGID ELEMENT WITH INITIAL VELOCITY

        PORDER: PROBLEM ORDER.
        ******
NUMBER OF ELEMENTS      1
NUMBER OF NODES      2
NUMBER OF VARIABLES AT EACH NODE      3
PROBLEM ORDER      6

        INTGTYP: INTEGRATOR PARAMETERS.
        ********

        INCMASS : MASS MATRIX UPDATE PARAMETERS
        ********
INCREMENT      MAX. RIGID      USER SPECIFIED
(# STEPS)        ANGLE         CONDITION LABEL
        1       .8750E-01

        TIMSPAN : ANALYSIS TIME RANGE.
        ********
START TIME      .0000E+00
STOP TIME      .1000E+01

        TIMSTEP: INTEGRATION TIME STEP SPECIFICATIONS
        ********
STEP SIZE      USER SPECIFIED
 (SEC)         CONDITION LABEL
  .1000E-02

        MATPROP: MATERIAL PROPERTIES
        ********
MAT. PROP   RIGIDITY      YOUNGS          BEAM X-SECT      MOMENT OF      DENSITY        MATERIAL          USER SPECIFIED
  ID #        TYPE        MODULUS            AREA           INERTIA      F*T**2/L**4      NAME            CONDITION LABEL
    1           3       .10500E+08       .31416E+01       .78540E+00      .25880E-03      ALUMINUM

NOTE ON RIGIDITY TYPE: 0 - FULLY FLEXIBLE 1 - AXIALLY RIGID, 2 - FLEXURALLY RIGID, 3 - FULLY RIGID
```

Figure 7.1-3b

GRID: GRID POINT LOCATIONS

G.P.NUMBER	X	Y
1	.00000E+00	.00000E+00
2	.12000E+02	.00000E+00

MEMBER: MEMBER DESCRIPTION AND CONNECTIVITY

MEMB NO	NODE A	NODE B	LENGTH	ANGLE	MATPROP ID #	CALABEL	NATURAL FREQUENCIES IN HZ (PERIOD SECS) BENDING (C-CMODE)	STRETCHING
1	1	2	.12000E+02	.00000E+00	1			

ADMASS: ADDITIONAL MASS MATRIX TERMS

NODE1	PARAM	NODE2	PARAM	MASS	USER DEFINED CONDITION LABEL
1	1	1	1	.10000E-02	
1	2	1	2	.10000E-02	
2	1	2	1	.10000E-02	
2	2	2	2	.10000E-02	

VEL: NON-ZERO INITIAL VELOCITIES SPECIFIED

NODE ID#	PARAMETER	COORD-SYST. OF MEMBER #	MAGNITUDE
1	3	0	-.3142E+01
2	3	0	-.3142E+01
1	2	0	.1885E+02
2	2	0	-.1885E+02

PRINT: PRINT OUTPUT SPECIFICATIONS

SOLUTION VECT. INC.	COORDINATE SYSTEM	INC.	FORCES VECTOR COORD.SYST	MASS MATRIX INCREMENT	STIFFNESS MATRIX INCREMENT	USER SPECIFIED CONDITION LABEL
100	0	100	LOCAL	0	0	

PLOT: POSTPROCESSOR OUTPUT FILE SPECIFICATION

INCREMENT	USER SPECIFIED CONDITION LABEL
10	

START TIME FOR PLOT OUTPUT .0000E+00
STOP TIME FOR PLOT OUTPUT .1000F+01

Figure 7.1-3c

```
                    ****************************************
                    *                                      *
                    *                                      *
                    *  PRELIMINARY COMPUTATIONS AND PRESETS *
                    *                                      *
                    *                                      *
                    ****************************************

MASS PROPERTIES OF TOTAL STRUCTURE
----------------------------------
CENTROIDAL COORDINATES ARE XBAR=  .49792907E+01 YBAR=  .10207093E+01
TOTAL MASS=  .11756530E-01
MOMENTS OF INERTIA ABOUT ORIGIN ARE IX=  .46831345E+00 IY=  .14400000E+00
MOMENTS OF INERTIA ABOUT C.G. ARE IXO=  .17682985E+00 IYO=  .13175149E+00
ROTATIONAL INERTIA ABOUT ORIGIN =  .61231345E+00
ROTATIONAL INERTIA ABOUT C.G. =  .30858134E+00

MATRIX BANDWIDTH AND CONSTRAINT SUMMARY
---------------------------------------
MAX. POSSIBLE NUMBER OF CONSTRAINTS FOR THIS RUN     3

PRESET DISPLACEMENTS AND VELOCITIES
-----------------------------------

                        GRID POINT TRANSLATION                          GRID POINT ROTATION
SEQ #  GP. #   X(DISP)      X(VEL)      Y(DISP)      Y(VEL)       PHI(DISP)     PHI(VEL)
    1      1  .00000E+00  .00000E+00  .00000E+00  .18850E+02   .00000E+00  -.31416E+01
    2      2  .00000E+00  .00000E+00  .00000E+00 -.18850E+02   .00000E+00  -.31416E+01

DYNAMIC DIRECT ACCESS MEMORY USAGE
----------------------------------
FIELD LENGTH FOR DYNAMIC STORAGE AREA 1       94
FIELD LENGTH FOR DYNAMIC STORAGE AREA 2      280
TOTAL DYNAMIC STORAGE      374

                    ********************************
                    *                              *
                    *                              *
                    *  TIMESTEP INTEGRATION DATA   *
                    *                              *
                    *                              *
                    ********************************

NOTES ON INTEGRATION PROCEDURE AND PRINTING SEQUENCES
-----------------------------------------------------

ZEROETH TIMESTEP: DISPLACEMENTS AND VELOCITIES AT THE ZEROETH TIMESTEP ARE SET EQUAL TO THEIR INITIAL VALUES AS
                  SPECIFIED IN THE INPUT DATA. ACCELERATIONS AT THIS STEP TIME ARE CALCULATED FROM THESE INITIAL VALUES.
                  USER DEFINED CONDITIONS AND Q-VARIABLES AT THE ZEROETH TIME STEP ARE EQUAL TO THEIR PRESET VALUES.

SUBSEQUENT TIMESTEPS: DISPLACEMENTS, VELOCITIES AND ACCELERATIONS AT SUBSEQUENT TIMESTEPS ARE COMPUTED FROM
                  THEIR VALUES AT THE PREVIOUS STEP TIME (OR TIMES) USING THE SPECIFIED INTEGRATION METHOD. USER DEFINED
                  CONDITIONS OR Q-VARIABLES (IF NEEDED) FROM THE PREVIOUS STEP TIME ARE ALSO USED IN THESE CALCULATIONS. WHEN A
                  PRINTOUT OF SOLUTION VECTORS IS TRIGGERED, THEN CONDITIONS AND Q-VARIABLES AT THE PREVIOUS STEP TIME AS WELL
                  AS AT THE PRESENT STEP TIME ARE GIVEN.
```

Figure 7.1-3d

INTEGRATION STEP NUMBER 0 STEP TIME .00000000E+00

INTERNAL FORCES (PRINTED IN THE LOCAL COORDINATES OF EACH MEMBER AT ENDS A AND B)

MEMB. NO.	AXIAL FORCES A	AXIAL FORCES B	SHEAR FORCES A	SHEAR FORCES B	BENDING MOMENTS A	BENDING MOMENTS B
1	.00000000E+00	.00000000E+00	.00000000E+00	.00000000E+00	.00000000E+00	.00000000E+00

SOLUTION VECTORS

SEQ #	GP. #	GRID POINT TRANSLATION X(DISP)	X(VEL)	X(ACCEL)	Y(DISP)	Y(VEL)	Y(ACCEL)	GRID POINT ROTATION PHI(DISP)	PHI(VEL)	PHI(ACCEL)
1	1	.00000F+00	.00000E+00	.59218E+02	.00000E+00	.18850E+02	.00000E+00	.00000F+00	-.31416E+01	.00000F+00
2	2	.00000E+00	.00000E+00	-.59218E+02	.00000E+00	-.18850E+02	.00000E+00	.00000F+00	-.31416E+01	.00000F+00

ABSOLUTE GRID POINT LOCATION IN GLOBAL CARTESIAN COORDINATES

SEQ #	GP. #	XLOC	YLOC
1	1	.00000F+00	.00000F+00
2	2	.12000E+02	.00000F+00

MEMBER DATA

MEMBER NO	ROTATION	MIDPLANE STRAIN
1	.00000E+00	.00000E+00

INTEGRATION STEP NUMBER 100 STEP TIME .10000000E+00

INTERNAL FORCES (PRINTED IN THE LOCAL COORDINATES OF EACH MEMBER AT ENDS A AND B)

MEMB. NO.	AXIAL FORCES A	AXIAL FORCES B	SHEAR FORCES A	SHEAR FORCES B	BENDING MOMENTS A	BENDING MOMENTS B
1	.00000000E+00	.00000000E+00	.00000000E+00	.00000000E+00	.00000000E+00	.00000000E+00

SOLUTION VECTORS

SEQ #	GP. #	GRID POINT TRANSLATION X(DISP)	X(VEL)	X(ACCEL)	Y(DISP)	Y(VEL)	Y(ACCEL)	GRID POINT ROTATION PHI(DISP)	PHI(VEL)	PHI(ACCEL)
1	1	.29366F+00	.58248E+01	.56319F+02	.18541E+01	.17927E+02	-.18299E+02	-.31416E+00	-.31416E+01	.42633F-13
2	2	-.29366E+00	-.58249E+01	-.56319E+02	-.18541E+01	-.17927E+02	.18299E+02	-.31416E+00	-.31416E+01	.42633F-13

ABSOLUTE GRID POINT LOCATION IN GLOBAL CARTESIAN COORDINATES

SEQ #	GP. #	XLOC	YLOC
1	1	.29366F+00	.18541F+01
2	2	.11706E+02	-.18541F+01

MEMBER DATA

MEMBER NO	ROTATION	MIDPLANE STRAIN
1	-.31416F+00	.12185F-08

Figure 7.1-3e

INTEGRATION STEP NUMBER 1000 STEP TIME .10000000E+01

INTERNAL FORCES (PRINTED IN THE LOCAL COORDINATES OF EACH MEMBER AT ENDS A AND B)

MEMB. NO.	AXIAL FORCES A	AXIAL FORCES B	SHEAR FORCES A	SHEAR FORCES B	BENDING MOMENTS A	BENDING MOMENTS B
1	.00000000E+00	.00000000E+00	.00000000E+00	.00000000E+00	.00000000E+00	.00000000E+00

SOLUTION VECTORS

SEQ #	GP. #	GRID POINT TRANSLATION X(DISP)	X(VEL)	X(ACCEL)	Y(DISP)	Y(VEL)	Y(ACCEL)	GRID POINT ROTATION PHI(DISP)	PHI(VEL)	PHI(ACCEL)
1	1	.12000E+02	-.11266E-03	-.59218E+02	.50421E-05	-.18849E+02	.35421E-03	-.31416E+01	-.31416E+01	.21684E-17
2	2	-.12000E+02	.11283E-03	.59218E+02	.76820E-04	.18850E+02	-.35421E-03	-.31416E+01	-.31416E+01	.21684E-17

ABSOLUTE GRID POINT LOCATION IN GLOBAL CARTESIAN COORDINATES

SEQ #	GP. #	XLOC	YLOC
1	1	.12000E+02	.50421E-05
2	2	-.30110E-04	.76820E-04

MEMBER DATA

MEMBER NO	ROTATION	MIDPLANE STRAIN
1	-.31416E+01	.12143E-07

INTEGRATION STEP NUMBER 1000 STEP TIME .10000000E+01

SOLUTION VECTORS

SEQ #	GP. #	GRID POINT TRANSLATION X(DISP)	X(VEL)	X(ACCEL)	Y(DISP)	Y(VEL)	Y(ACCEL)	GRID POINT ROTATION PHI(DISP)	PHI(VEL)	PHI(ACCEL)
1	1	.12000E+02	-.11266E-03	-.59218E+02	.50421E-05	-.18849E+02	.35421E-03	-.31416E+01	-.31416E+01	.21684E-17
2	2	-.12000E+02	.11283E-03	.59218E+02	.76820E-04	.18850E+02	-.35421E-03	-.31416E+01	-.31416E+01	.21684E-17

ABSOLUTE GRID POINT LOCATION IN GLOBAL CARTESIAN COORDINATES

SEQ #	GP. #	XLOC	YLOC
1	1	.12000E+02	.50421E-05
2	2	-.30110E-04	.76820E-04

MEMBER DATA

MEMBER NO	ROTATION	MIDPLANE STRAIN
1	-.31416E+01	.12143E-07

TIMING STATISTICS FOLLOW:

CPU TIME FOR SUBS THETA,FORCES,FSUM,PROD,VELOC 11.209

CPU TIME SPENT ASSEMBLING MASS MATRIX 18.266

CPU TIME SPENT INVERTING MASS MATRIX 2.900

TOTAL TIME (EXCLUDING DATA INPUT) 32.375

TOTAL NUMBER OF NODAL VARIABLES 6

NUMBER OF TIME STEPS 999

Figure 7.1-3f

Problem 7.2 - Flexible Boom With Rigid Hub And Applied Torque

This sample problem involves the response of a flexible boom with a rigid central hub subject to an applied torque as shown in Figure 7.2-1. An antisymmetric boundary condition is used such that only one-half the structure is modelled. This sample problem demonstrates the application of external loads, boundary constraints, and condition labels.

Input Data

Figure 7.2-2 contains the input data for Problem 7.2. Condition labels are used throughout the problem and the user is referred to Section 5.16 for a complete description of condition labels. The condition labels are written as,

```
CL1:T.GT.1.0
CL2:T.GT.5.0
```

The first condition label (CL1) states that whenever the analysis time, T, exceeds the value of 1 second, the condition is "TRUE" and any commands which are dependent on this label are activated. Similarly for the second condition label (CL2) is "TRUE" whenever the time becomes greater than or equal to 5 seconds. The above implies that whenever a condition label is not "TRUE" it is "FALSE", thus any commands dependent on it are not active. In this example, condition label 1 is used in the TIMSTEP, PRINT and PLOTINC commands and condition label 2 is used in the STOP command.

In addition to the required and optional commands used and discussed in Problem 7.1, the following optional commands are used in the current problem,

- PLOTLIMIT
- INTGTYP
- STOP
- APPLOAD
- LOAD
- SDFC
- SCALE

The PLOTLIMIT command indicates a specific time duration for which the results are stored for plotting with the postprocessor. This command is used in conjunction with the PLOTINC command such that the user can obtain a large amount of data for plotting, but only during a specified portion of the analysis.

The INTGTYP command specifies the algorithm used in the transient analysis. In this example, the integration procedure used is central differences (INTGTYP=1).

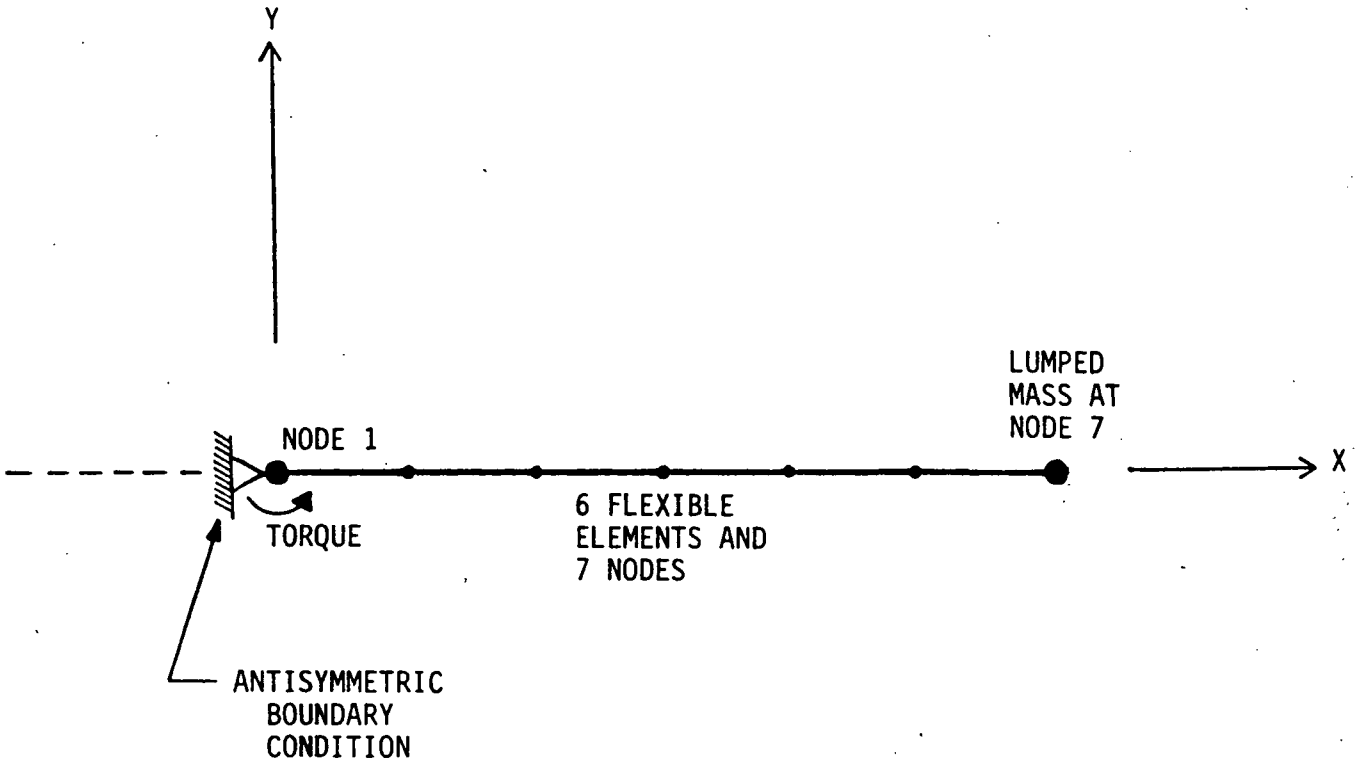


Figure 7.2-1 Flexible Boom With Rigid Hub And Applied Torque

```
$
$TITLE: DATA CASE TITLE (UP TO 80 CHARACTERS)
TITLE: T2S - SECOND TEST CASE FOR LATDYN USER'S GUIDE
$
$SUBTITLE:DATA CASE SUBTITLE (UP TO 80 CHARACTERS)
SUBTITLE: FLEXIBLE BOOM WITH RIGID CENTRAL HUB AND APPLIED TORQUE
$
$CONDITION LABELS DEFINITION
CL1:T.GT.1.0  $FIRST CONDITION TRUE WHEN TIME EXCEEDS 1 SECOND
CL2:T.GE.5.0  $CONDITION TRUE WHEN TIME REACHES 5 SECONDS
$
$DEFINE: @N=CONSTANT
DEFINE: @1=11.2E06 $DEFINE VALUE OF YOUNG'S MODULUS
DEFINE: @99=3.3376E09$UP TO 99 CONSTANTS CAN BE DEFINED
$
$PORDER: NELM,NODES
PORDER:  6,  7 $DEFINE UP TO 6 ELEMENTS AND 7 NODES
$
$TIMSPAN: START,STOP
TIMSPAN:  0.0, 10.0  $ANALYSIS INTENDED TO RUN FOR 10 SECONDS
$
$TIMSTEP:DELTA T ?CONDITION LABEL
TIMSTEP: 3.0E-5  $ORIGINAL TIME STEP
         3.0E-4 ?1 $MODIFY TIME STEP IF CONDITION ONE IS TRUE
$
$MATPROP:MATNAME,MTYPE, E   ,  A    ,   I   ,  RHO ? CONDITION LABEL
MATPROP:  1,     0,  @1, 48.440, 193.60, 2.588E-4 $DEFINITION OF MATERIAL
$AND STRUCTURAL PROPERTIES WITH MATERIAL NAME 1
$
$GRID:NST,# NODES, X1  , Y1   ,X2   , Y2  ,INC
GRID: 1,   7,  0.0,    0.0,  100.0, 0.0,  1 $SPECIFY 7 NODES EQUALLY SPACED
$FROM ZERO TO 100 ALONG THE X-AXIS
$
$MEMBER:IMEMB  ,N1  ,N2  ,NE   ,MAT
MEMBER:  ,  1,   2,  6,       1 $SPECIFY,6 ELEMENTS CONNECTING THE 7 NODES
$
$
$PRINT:DVA INC, COORD,INTF INC,COORD,MM INC,SM INC ?CONDITION LABEL
PRINT: 16666,   0,     16666,   L,    0,     0  $PRINT RESULTS EVERY 16666
       1666,   0,      1666,   L,    0,     0 ?1  $DECREASE PRINT FREQUENCY
$WHEN THE TIME STEP IS INCREASED -IF CONDITION 1 IS TRUE
$
$PLOTINC: PLOT INC  ? CONDITION LABEL
PLOTINC:  1666  $STORE PLOTTING INFO. 10 TIMES AS OFTEN AS PRINTOUT
           166 ?1  $CHANGE PLOTINC CONSISTENT WITH PRINT INCREMENT
$
$PLOTLIMIT:LOWER TIME LIMIT,UPPER TIME LIMIT
PLOTLIMIT:  0.0,    5.0   $ONLY DESIRE PLOTS FOR FIRST 5 SECONDS
$
```

Figure 7.2-2 Input Data For Problem 7.2

```
$BATCH PLOTTING COMMANDS
SNAPSHOT,0.0,5.0,400,0
PLOTY,D,7
PLOTPH1,D,1
$
$INCMASS: NSTEPS,ANGLE?CONDITION LABEL
INCMASS:    0, 0.0875 $NO UPDATING OF THE MASS MATRIX IS REQUIRED
$
$INTGTYP: INTG,(BETA)
INTGTYP:1
$
$STOP: ? CONDITION LABEL
STOP: ?2  $STOP ANALYSIS WHEN CONDITION 2 IS TRUE
$
$
$ADMASS: NDE1, IDIR1, NDE2, IDIR2, MASS ?CONDITION LABEL
ADMASS:   7,     1,     7,     1,     248.0  $LUMPED MASS AT NODE 7 IN THE
          7,     2,     7,     2,     248.0  $X AND Y DIRECTIONS
          1,     3,     1,     3,  @99   $ADD ROTATIONAL INERTIA TO NODE 1
$
$APPLOAD:LOAD ID, IMEMB ,DIR   ,SCALE ,TSTART,TSTOP,GRID PNTS    ? CONDITION
APPLOAD:    1,   ,  TORQUE, 10.0,  0.0,    10.0,    1   $APPLY TORQUE TO NODE 1
$
$LOAD:LOAD ID, AMPLITUDE
LOAD:    1,    1.0904E07  $SPECIFIES VALUE OF TORQUE USED BY APPLOAD
$
$
$SDFC:GRID NO,CON DIR,CONSTANT DOF  ? CONDITION LABEL
SDFC:     1,     1,     0.0 $CONSTRAIN MOTION OF NODE 1 IN X DIRECTION
          1,     2,     0.0 $CONSTRAIN MOTION OF NODE 1 IN Y DIRECTION
$
$
$SCALE: SCALEX,SCALEY, ROT
SCALE: 20.0,   1.0,   0.0  $SCALE GEOMETRY IN X DIRECTION BY 20
$HAS THE NET EFFECT OF MAKING THE TOTAL STRUCTURE 2000 UNITS LONG
```

Figure 7.2-2 continued

As mentioned earlier, the STOP command can be used to terminate program execution prior to the stop time specified on the TIMSPAN command. The program can be stopped anytime during execution once the condition label associated with the STOP command becomes true. Only one STOP command can be included in the data set. In this example, the program is stopped once condition label 1 becomes true.

Using the APPLOAD command the user specifies the direction, time duration and location of an externally applied load. The user has the option of specifying the direction of an applied load in terms of the local coordinate system of any member, if not specified then the direction is assumed to be relative to the global coordinate system. This command is required when a load is to be applied to the structure (with the exception of GRAVITY) and must be used in conjunction with any of the LOAD, LOADSIN, or LOADTIM commands. In this example, APPLOAD is used along with the LOAD command to apply a torque at node 1 which is to act for 10 seconds. The scale factor of 10 is applied to the magnitude of the load specified on the LOAD command. The application of loads through the APPLOAD command can be controlled with condition labels to turn loads on and off during the analysis.

The LOAD command specifies a constant load with a given magnitude. Each LOAD command has a load identification number which is also used by the APPLOAD command to designate the location, direction and scaling of the load. For this case, the load identified as "1" is a torque at node 1 with a magnitude which will be scaled by 10 when it is applied via the APPLOAD command.

Single degree-of-freedom and boundary constraints are specified with the SDFC command. The user can specify that the motion of a grid point be fixed at a given value. In this example, the x and y motions of grid point 1 are constrained to zero although it is allowed to rotate. Thus, grid point 1 has effectively been pinned to simulate the antisymmetric boundary condition desired. It is possible to make the SDFC constraints conditional such that they can be imposed or released throughout the analysis.

By using the SCALE command the user can scale the dimensions of the structure in the global x or y directions as well as rotate the entire structure. This command affects <u>all</u> of the coordinates of the grid points defined by the GRID command. This problem uses SCALE to increase the length of the structure by 20 in the x direction while maintaining the original dimensions in the other directions.

Sample Plots

Figures 7.2-3 through 7.2-5 show the plots generated (via the postprocessor) by the plotting commands SNAPSHOT, PLOTY, and PLOTPHI. These commands are included in the input data set for this sample problem as described in Section 5.19.4.

The command used to produce Figure 7.2-3 was,

```
SNAPSHOT,0.0,5.0,400,0
```

and the result is a series of structural deformation plots overlaid.

Figure 7.2-4 depicts the time history of the displacement of node 7 parallel to the global y axis and was produced by the command,

```
PLOTY,D,7
```

Finally, a time history of the rotational motion of node 1 is given in Figure 7.2-5. The command used to obtain this plot was,

```
PLOTPHI,D,1
```

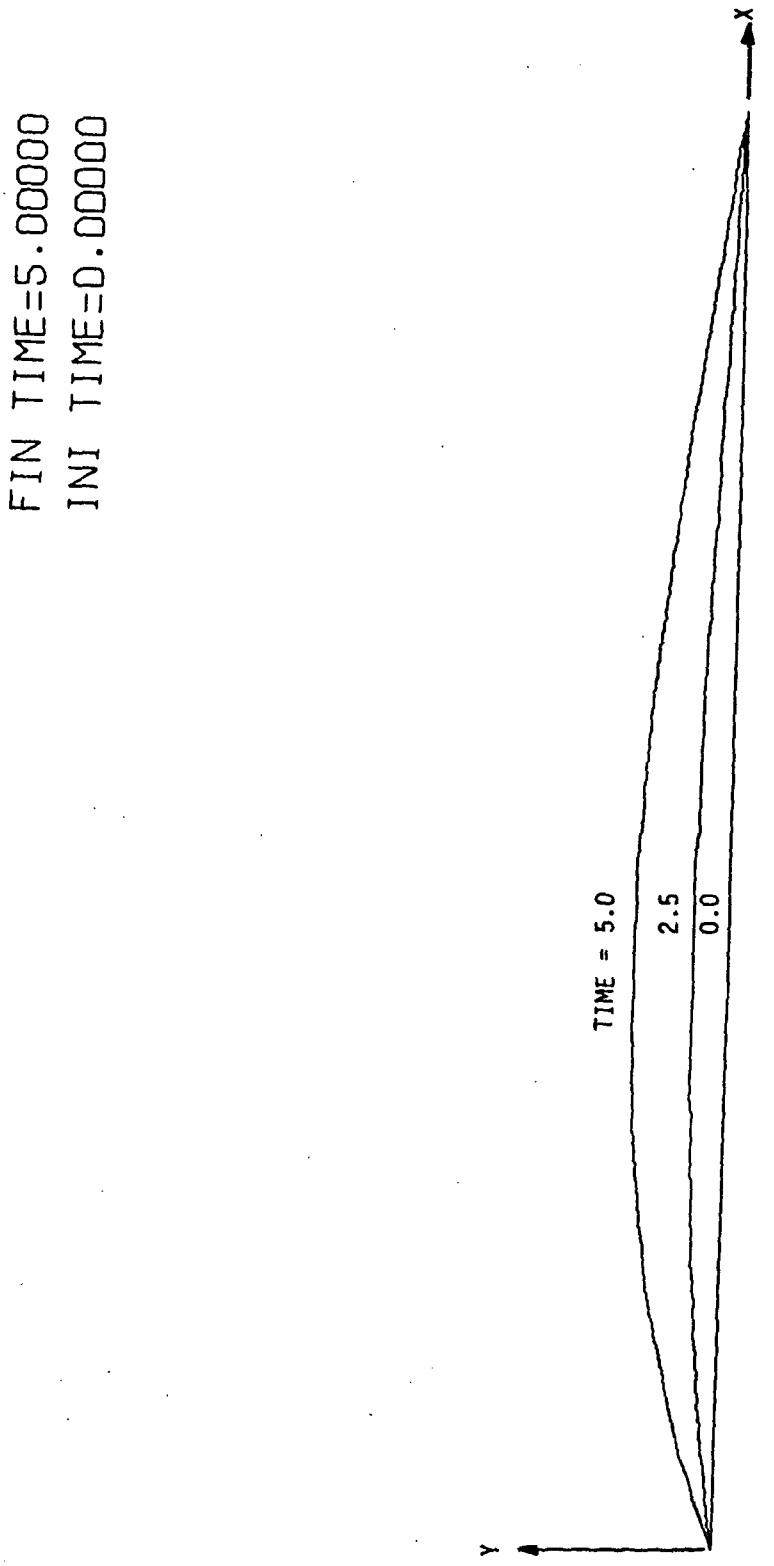


Figure 7.2-3 Plot Created by SNAPSHOT Command

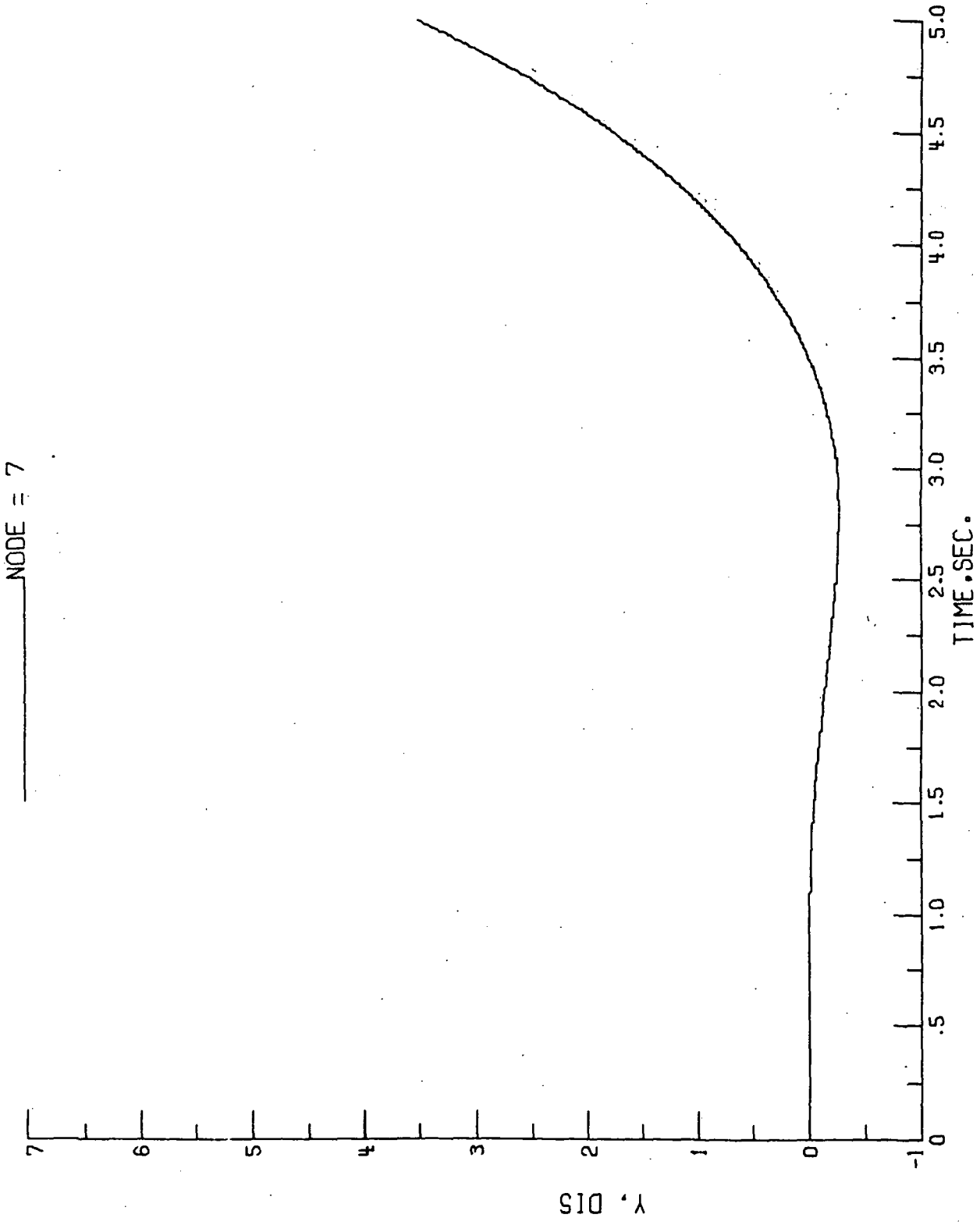


Figure 7.2-4 Plot Created by PLOTY Command

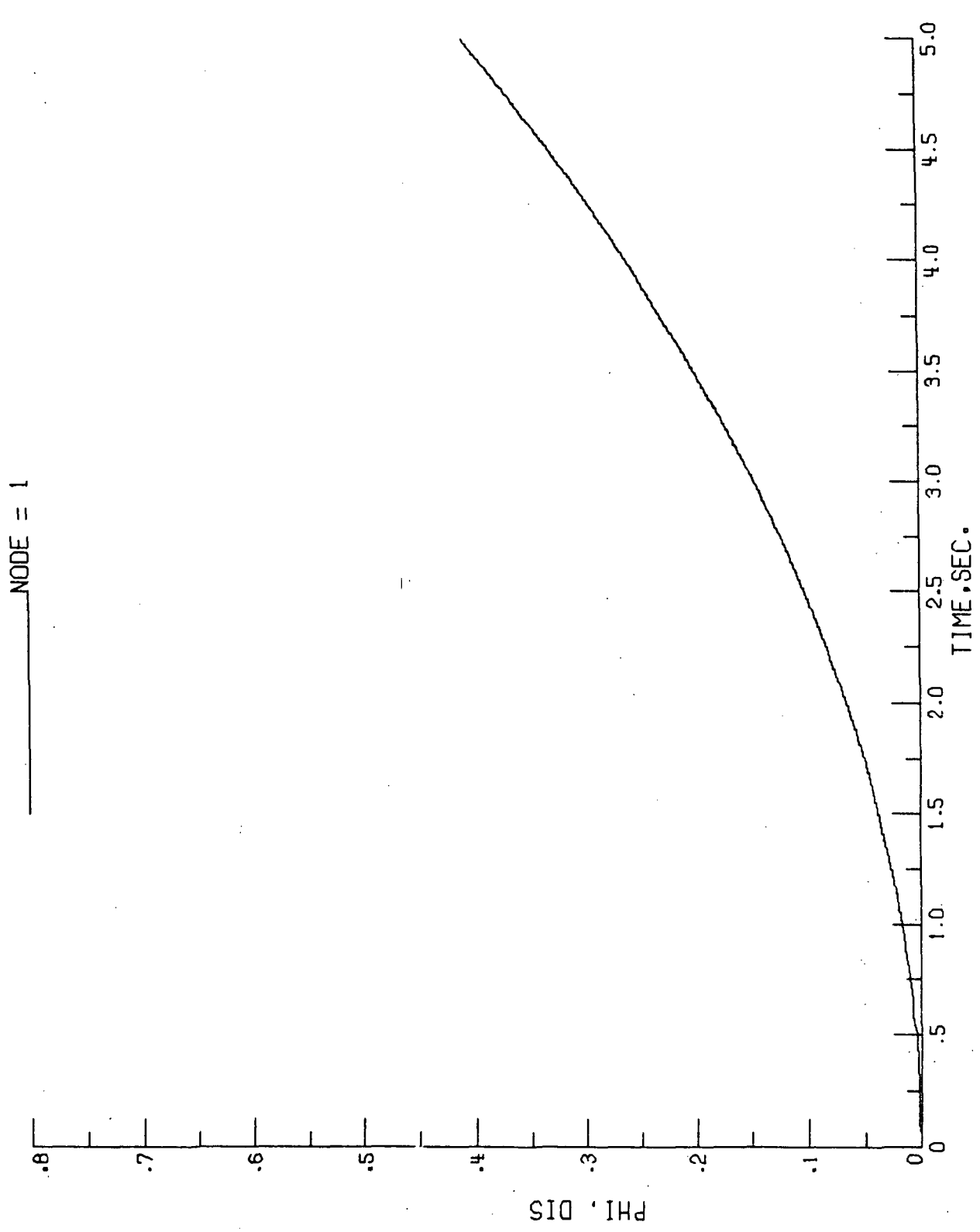


Figure 7.2-5 Plot Created by PLOTPHI Command

Problem 7.3 - Unfolding Deployment of One Bay

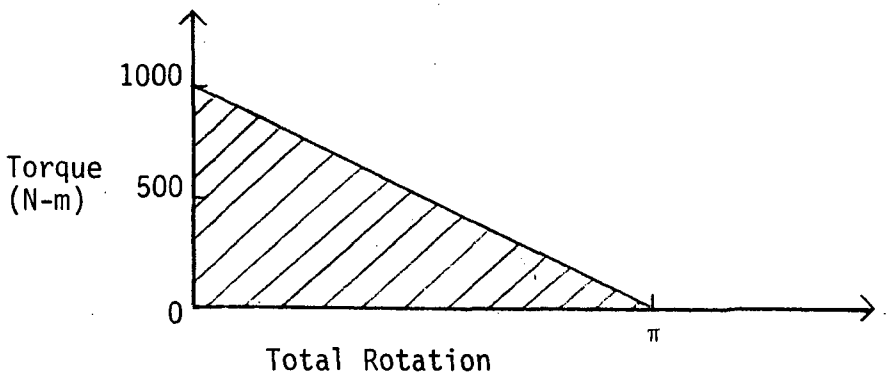
The problem modelled in this example is the unfolding deployment and lock-up of two flexible beams as depicted in Figure 7.3-1. The application of a rotational spring at a pin joint is demonstrated as well as the method for locking a joint. In addition, user defined variables are introduced.

Input Data

The data set for this problem is shown in Figure 7.3-2. All of the required commands and most of the optional commands which were discussed in the previous problems have been used in this example. The following optional commands are also used,

SPRING
PIN
ROTLOCK
SET
RESET

The SPRING command is used to define a spring connected between two grid points. Springs can be rotational or extensional and can act in a fixed direction or follow the line of action between two grid points. In addition to the grid point numbers and the direction, the user specifies the spring constant and any initial tension/compression (positive for tension and negative for compression). In this example, a rotational spring (direction=3) is defined between points 2 and 3 with a spring constant of 1000.0 and an initial compression of π radians. Thus, the torque provided by the spring is a linearly decreasing function of the rotation of the two nodes such that the torque reaches zero when the total rotation of the nodes is π (see sketch below).



Springs can be added or removed at anytime during the analysis by using a condition label. Thus, it is possible to synchronize the deployment of a multi-bay truss by activating the deployment springs at prerarranged times.

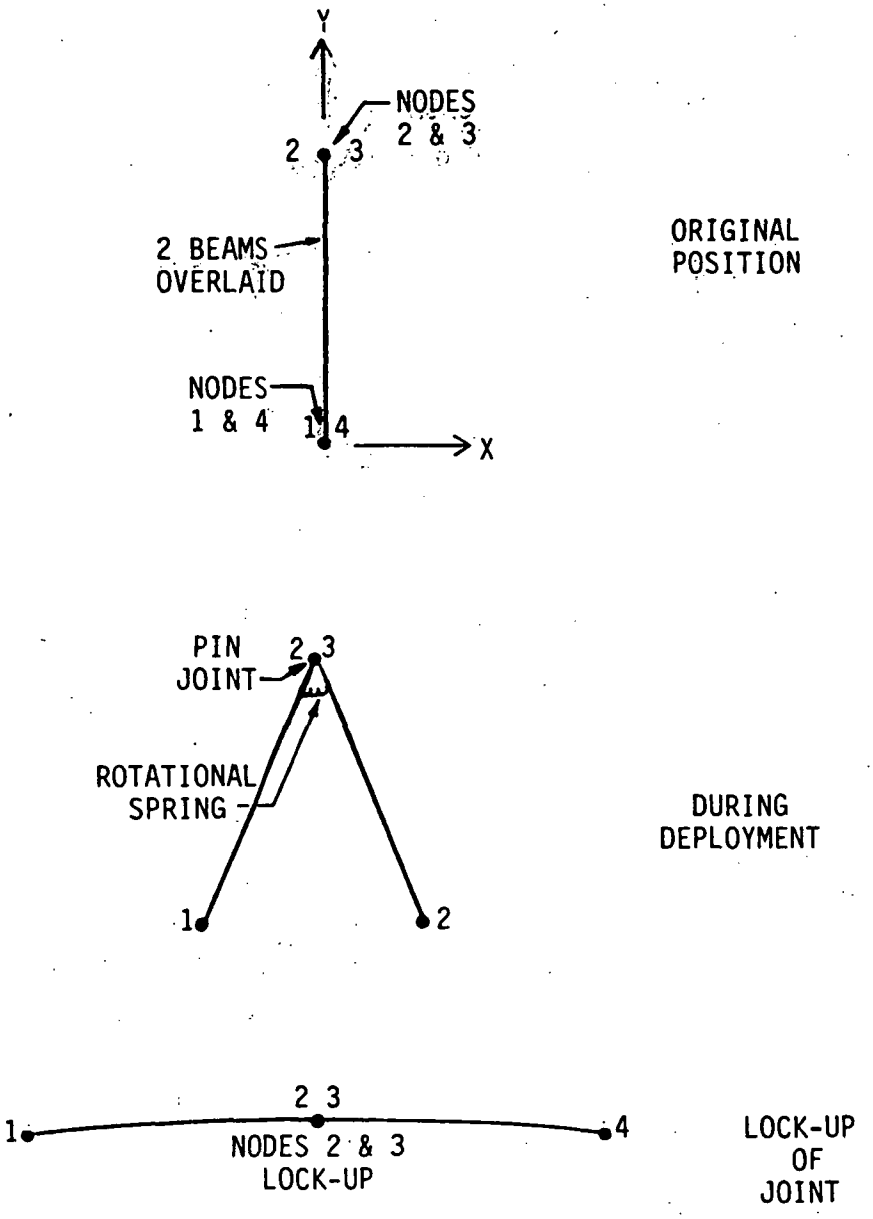


Figure 7.3-1 Unfolding Deployment Of One Bay

```
$
$
$TITLE: DATA CASE TITLE (UP TO 80 CHARACTERS)
TITLE:T1S - THIRD TEST CASE FOR LATDYN USER'S GUIDE
$
$SUBTITLE:DATA CASE SUBTITLE (UP TO 80 CHARACTERS)
SUBTITLE:UNFOLDING DEPLOYMENT OF ONE BAY
$
$DEFINE: @N=CONSTANT
DEFINE: @1=3.14159265  $DEFINE PI
$
$PORDER: NELM,NODES
PORDER:2, 4  $MAXIMUM OF 2 ELEMENTS AND FOUR NODES
$
$TIMSPAN: START,STOP
TIMSPAN: 0.0, 5.0  $FIVE SECOND ANALYSIS
$
$TIMESTEP:DELTA T ?CONDITION LABEL
TIMSTEP: .25E-3   $TIME STEP OF 0.00025 SECONDS
$
$MATPROP:MATNAME,MTYPE, E   ,  A    ,   I   ,  RHO ? CONDITION LABEL
MATPROP:TYPE1,    0, 70.E9, .0003688, 5.7625E-6, 2712.0  $DEFINITION
$OF MATERIAL AND STRUCTURAL PROPERTIES WITH NAME "TYPE1"
$
$GRID:NST,# NODES, X1   , Y1    ,X2    , Y2  ,INC
GRID:  1,    2,   0,    0,    0,     25,   1 $DEFINE TWO NODES 1&2
       3,    2,   0,  25.,    0,      0,   1 $DEFINE TWO NODES 3&4
$
$MEMBER:IMEMB   ,N1   ,N2   ,NE    ,MAT
MEMBER: ,1,    2,    1,  TYPE1 $DEFINE MEMBER FROM NODE 1 TO NODE 2
        ,3,    4,    1,  TYPE1 $DEFINE MEMBER FROM NODE 3 TO NODE 4
$
$
$
$PRINT:DVA INC, COORD,INTF INC,COORD,MM INC,SM INC ?CONDITION LABEL
PRINT:  1000,      0,    1000,     L,    0,     0 $PRINT EVERY 1000
$
$PLOTINC: PLOT INC ?CONDITION LABEL
PLOTINC:100 $PLOT EVERY 100 STEPS
$
$BATCH PLOTTING COMMANDS
SNAPSHOT,0.0,3.0,12,0,A
PLOTROT,2
PLOTPHI,D,3
PLOT0,3,5
PLOT0,5
$
$INCMASS: NSTEPS,ANGLE ?CONDITION LABEL
INCMASS:200000, .0875 $INCREMENT MASS MATRIX EVERY 200K STEPS
$
```

Figure 7.3-2 Input Data For Problem 7.3

```
$
$SPRING: NDE1,NDE2,IDIR,SPNGK,PSSPNG ? CONDITION LABEL
SPRING:    2,    3,    3,   1000., -@1 $DEFINE ROTATIONAL SPRING BETWEEN
$NODES 2 AND 3 WITH MAGNITUDE OF 1000
$
$
$PIN:GRIDPT1,GRIDPT2 ?CONDITION LABEL
PIN:   2,      3  $DEFINE PIN JOINT BETWEEN NODES 2 AND 3
$
$CLI: CONDITION LABEL DEFINITION
CL1: ABS(PHI(D,2)-PHI(D,3)).GE.Q2 $CONDITION TRUE WHEN NODES
$ 2 AND 3 HAVE ROTATED THROUGH A SUM OF PI RADIANS
CL2: NTSTEP.GE.1 $CONDITION 2 TRUE AT FIRST TIME STEP
$
$ROTLOCK:GRD PT1, GRD PT2 ?CONDITION LABEL
ROTLOCK:     2,       3 ?1  $ROTATIONAL LOCKUP OF NODES 2 AND 3 WHEN
$CONDITION 1 IS TRUE
$
$
$
$SET: QN=FORTRAN ?CONDITION LABEL
SET: Q1=ABS(YLOC(1)-YLOC(2)) ?2 $DEFINE ELEMENT LENGTH AT TIME ZERO
SET: Q2=4.*ATAN(1.0) $ALTERNATE DEFINITION OF PI
$
$RESET: QN=FORTRAN ?CONDITION LABEL
RESET: Q3=ABS(PHI(D,2)-PHI(D,3)) $TOTAL ROTATION BETWEEN NODES 2 AND 3
RESET: Q4=ABS(YLOC(1)-YLOC(2)) ?2 $RELATIVE DISPLACEMENT BETWEEN NODES 2 AND 3
RESET: Q5=Q4/Q1 ?2 $CHANGE IN DISPLACEMENT NORMALIZED BY ORIGINAL LENGTH
```

Figure 7.3-2 continued

The PIN command is a convenient method of creating a pin joint between two grid points. This command is used in lieu of two MDFC commands to constrain the translational degrees-of-freedom of the points together while allowing them to rotate independently. In this example points 2 and 3 are pinned together. It is also possible to create or remove pins during the analysis with condition labels.

Using the ROTLOCK command, the user can simulate the rotational lock-up of a joint. This option is used when it is desired to lock a joint without modelling the details of the joint as discussed in Section 5.17.2. In this example, the pin joint between grid points 2 and 3 is locked up when condition label 1 is "TRUE". From the CL1 command, condition label 1 is "TRUE" when the total rotation of both nodes exceeds the user defined variable Q2, which is defined as π (see discussion of SET command below).

The SET and RESET commands are user defined commands which allow the specification of variables in the form of FORTRAN expressions which can involve any of the physical quantities measured by the program at any time during the execution (the user is referred to Section 5.17 for a more complete description of these commands). In this example, the SET command is used to compute the length of one element as the absolute value of the difference in the y locations of nodes 1 and 2 in the global coordinate system. This command is activated when condition label 2 is "TRUE", i.e., at the first time step (see CL2 command in Figure 7.3-2). Note that once activated, the value of the Q-variable specified by a SET command remains constant, and can then be used elsewhere in the data set. The second SET command is used to define the value of π which is used in the CL1 command.

Also used in this example are three RESET commands. Once activated, the variables specified by each of the RESET commands are recomputed at every time step, in contrast to the SET command whose Q-variable is evaluated only once. The first RESET command computes the total rotation between points 2 and 3 at every time step (note that the variable Q3 could replace the lefthand side of the expression in the CL1 command). The second RESET command specifies the computation of the relative displacements of points 1 and 2 as was done in the first SET command. Finally, the third RESET command allows the calculation of the change in the relative displacements of points 1 and 2 normalized by the original length by dividing Q4 by Q1.

Sample Plots

Plotting is requested in this problem due to the presence of the SNAPSHOT, PLOTPHI, and PLOTQ commands in the input data set. Figures 7.3-3 through 7.3-5 contain the plots generated by these commands.

The deployment sequence of the two beams is depicted in Figure 7.3-3 by the series of overlaid structural deformation plots at various times. The command which generated this plot was,

```
SNAPSHOT,0.0,3.0,12,0,A
```

Figure 7.3-4 is a time history of the rotation of node 3 which was produced by the command,

```
PLOTPHI,D,3
```

The magnitudes of the two user defined variables Q3 and Q5 are depicted in Figure 7.3-5. The command used to plot the Q-variables was,

```
PLOTQ,3,5
```

Note that the joint lock-up time is the time at which the user defined variable Q3 reaches π radians.

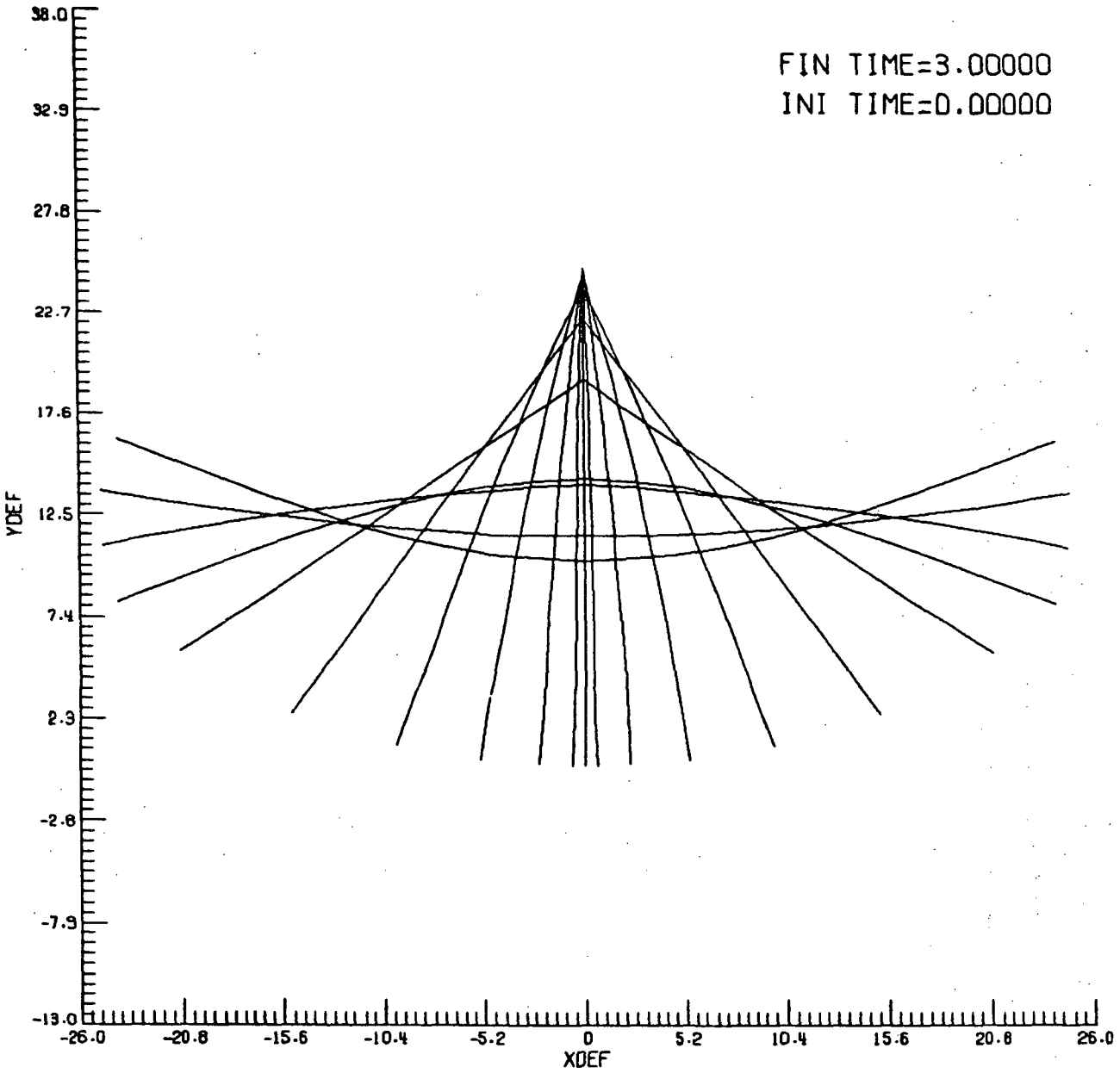


Figure 7.3-3 Plot Created by SNAPSHOT Command

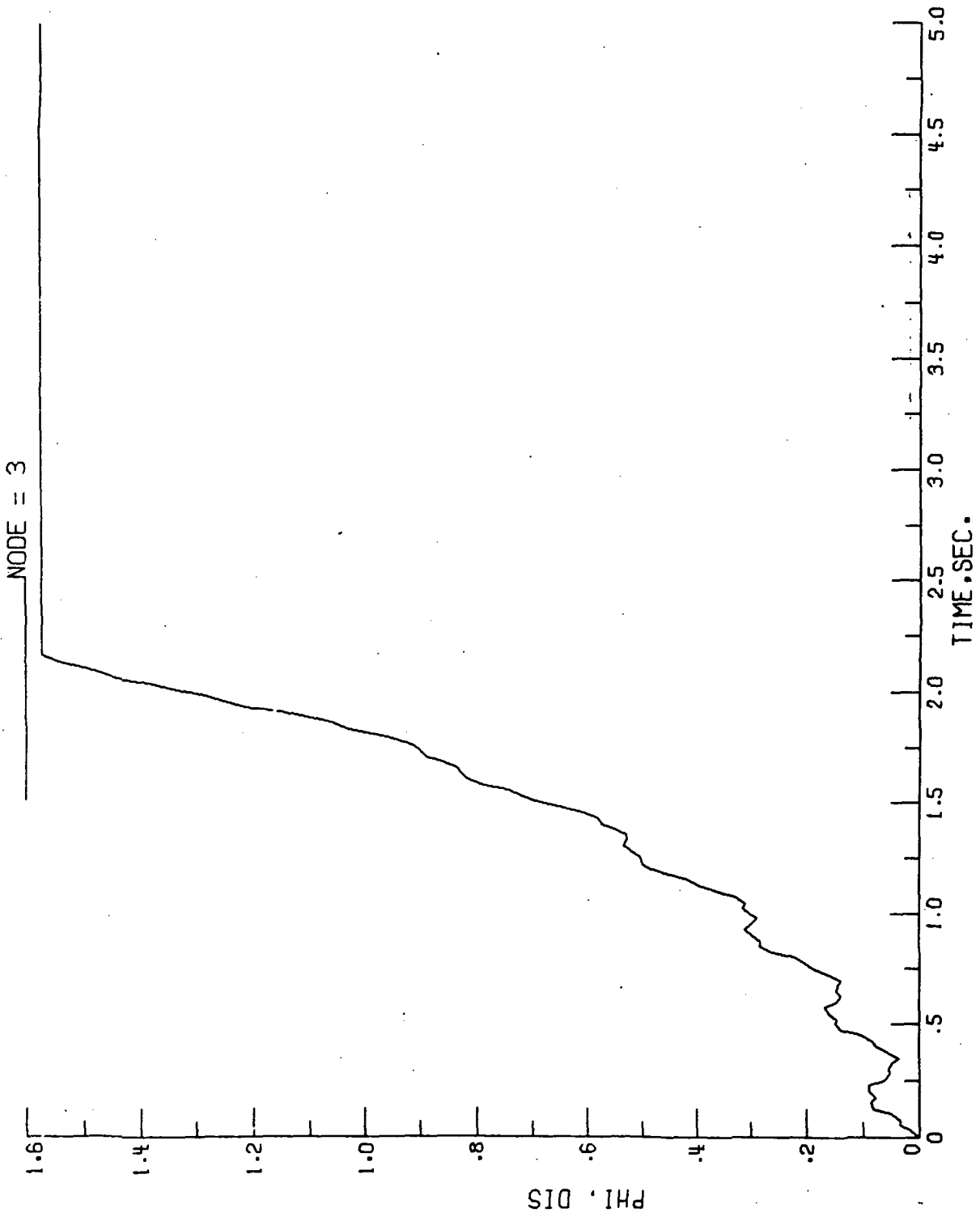


Figure 7.3-4 Plot Created by PLOTPHI Command

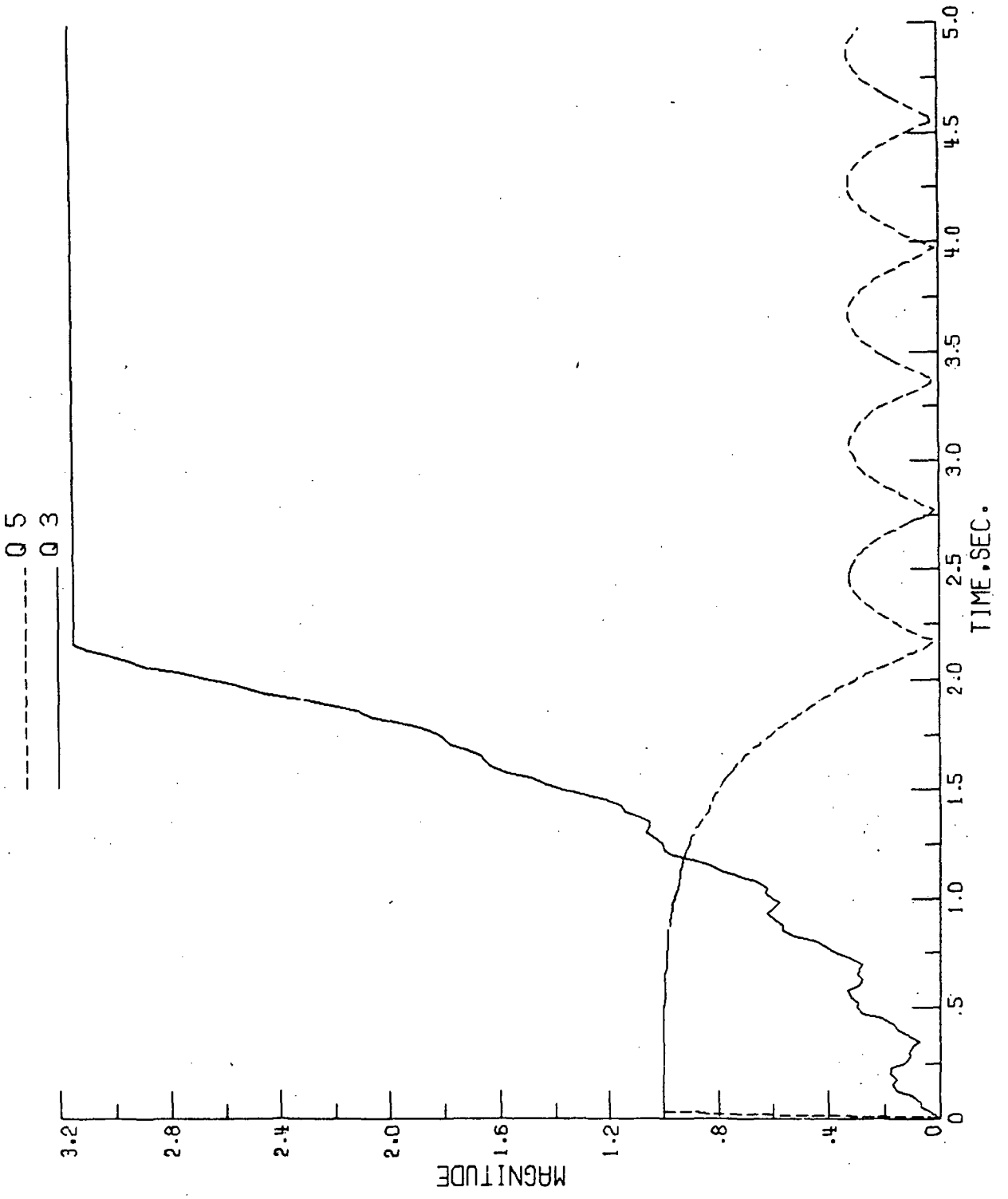


Figure 7.3-5 Plot Created by PLOTQ Command

Problem 7.4 - Rotating Rigid Beam With Control Forces

Problem 7.4 involves a rigid beam element with an initial rotational velocity of $-\pi$ rad/sec as in Problem 7.1. Q-variables are used to construct control forces normal to the beam and proportional to its velocity which act to slow it down first to $-\pi/2$ rad/sec and then to $-\pi/4$ rad/sec. Of interest in this problem is the use of condition labels to activate and deactivate the control forces as well as the user defined variables to provide the appropriate direction and magnitude of the forces.

Input Data

The basic model for this problem is shown in Figure 7.4-1 and the input data is given in Figure 7.4-2. There are five condition labels specified in the data. These are used to specify the instances at which the control forces are activated. Condition label 1 is "TRUE" when grid point 1 has rotated through an angle equal to the value specified by the Q-variable Q5, namely 10 degrees. Condition label 2 becomes "TRUE" when CL1 is "TRUE" and the rotational velocity of point 1 is greater than or equal to $\pi/2$ (note the use of the logical .AND. in the condition label). Effectively, CL2 is used to turn on the control force when the rotational velocity of the beam is greater than that desired. Once the new velocity state is reached, CL2 becomes "FALSE" and the control force is removed.

Condition labels 3, 4 and 5 are used in a similar manner to allow the beam to reach the second rotational velocity state specified by Q6 and Q7, where Q6 and Q7 are defined through the SET command.

The SET command is used to define the desired rotational velocity states specified in the condition labels as Q6, Q7 and Q8. These values are computed once at the first time step and their values remain constant throughout the analysis.

Using the RESET command, the angle of the velocity vector at grid points 1 and 2 is computed at each time step as the variable names Q1 and Q3. These angles are used in the APPLOAD command to specify the direction of the control forces normal to the beam. In addition, Q2 and Q4 are computed as the magnitude of the velocities of points 1 and 2 multiplied by a scale factor of 0.01, and are the scale factors for the control forces specified in the APPLOAD command.

The control forces are specified via the APPLOAD and LOAD commands. The APPLOAD commands specify the direction, scale factor, time duration and location of the control forces. The first two APPLOAD commands are activated when condition label 2 is "TRUE" and deactivated when the desired velocity state specified on the CL2 command is reached. Likewise, the next two pairs of APPLOAD commands specify the application of the control forces

when conditions 4 and 5 respectively are "TRUE". Note that the direction of the control forces is given by the angle of the velocity vector normal to the beam (Q1 and Q3), and the scale factor is proportional to the magnitude of the velocity of the beam (Q2 and Q4). Each APPLOAD command is assigned a load identification number which is referenced by a particular type of load. In this example, the load type is a constant load of amplitude 1.0 given by the LOAD command. Thus, the scale factor used in the APPLOAD command (which varies with the magnitude of the beams velocity) is used to scale the amplitude of the LOAD command thereby creating an external load whose magnitude can vary at each time step.

Sample Plots

In this example problem, 5 plots are requested through the plotting commands SNAPSHOT, PLOTC, PLOTMAG, PLOTQ, and PLOTROT which are in the input data set. The plots are shown in Figures 7.4-3 through 7.4-7.

A series of overlaid plots of the beam at various times are shown in Figure 7.4-3. These results were produced by the command,

```
SNAPSHOT,0.0,4.0,55,0
```

The logical status of the 5 condition labels during the analysis is shown in Figure 7.4-4, and the command which generated this plot was,

```
PLOTC,1,2,3,4,5
```

Figure 7.4-5 shows a time history of the magnitude of the velocity vector at node 1 which was created by the command,

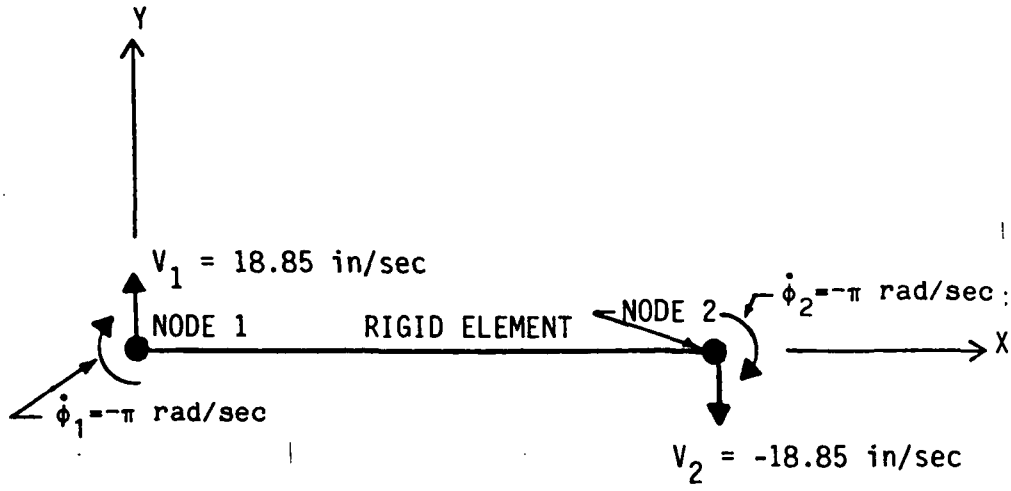
```
PLOTMAG,V,1
```

The magnitude of the user defined Q-variable Q2 during the analysis is presented in Figure 7.4-6. The command used to produce this plot was,

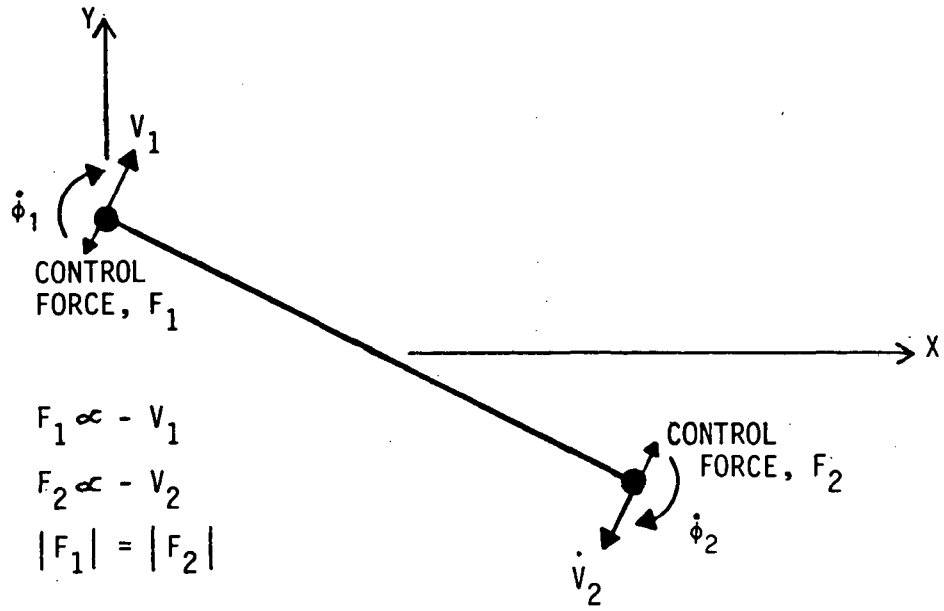
```
PLOTQ,2
```

Finally, the rotation of the beam element is depicted in Figure 7.4-7 and this corresponds to the command,

```
PLOTROT,1
```



(a) INITIAL CONDITIONS



(b) APPLICATION OF CONTROL FORCES

Figure 7.4-1 Rotating Rigid Beam With Control Forces

```
$
$THIS PROBLEM IS THE SAME AS THE FIRST PROBLEM BUT IT HAS CONTROL
$FORCES WHICH ARE PROPORTIONAL TO THE VELOCITY
$
$TITLE: DATA CASE TITLE ( UP TO 80 CHARACTERS)
TITLE: T9SA - FOURTH TEST CASE FOR LATDYN USER'S GUIDE
$
$SUBTITLE:DATA CASE SUBTITLE(UP TO 80 CHARACTERS)
SUBTITLE: ONE RIGID ELEMENT WITH INITIAL VELOCITY AND CONTROL FORCES
$
$DEFINE: @N=CONSTANT
DEFINE: @1=.0002588      $DEFINE MATERIAL DENSITY
DEFINE: @2=3.141592654    $ DEFINE PI
        @3= 1.0E-3  $DEFINE VALUE OF CONCENTRATED MASS
$
$CLI: CONDITION LABEL DEFINITION
CL1: ABS(PHI(D,1)).GE.05 $TRUE WHEN ROTATION OF NODE 1 = 05
CL2: CL1.AND.ABS(PHI(V,1)).GE.PI/2.0 $TRUE WHEN CL1 IS TRUE AND
$THE ROTATIONAL VELOCITY OF NODE 1 IS GREATER THAN PI/2
CL3: ABS(PHI(D,1)).GE.06 $TRUE WHEN NODE 1 ROTATES GREATER THAN 06
CL4: CL3.AND.ABS(PHI(V,1)).GE.PI/4.0 $TRUE WHEN CL3 IS TRUE AND
$THE ROTATIONAL VELOCITY OF NODE 1 IS GREATER THAN PI/4
CL5: ABS(PHI(D,1)).GE.07 $TRUE WHEN NODE 1 ROTATES GREATER THAN 07
$
$PORDER: NELM,NODES
PORDER:  1,   2       $ONE BEAM ELEMENT AND TWO NODES ARE DEFINED
$
$TIMSPAN: START,STOP
TIMSPAN: 0.0,    4.0  $ANALYSIS TO RUN FOR 4 SECOND
$
$TIMSTEP:DELTA T ?CONDITION LABEL
TIMSTEP: 1.0E-3       $INTEGRATION TIME STEP IS .001 SECONDS
$
$MATPROP:MATNAME,MTYPE, E   ,  A    ,   I   ,  RHO ? CONDITION LABEL
MATPROP: ALUMINUM, 3, 10.5E06, @2, .78540, @1  $ USING RIGID MEMBER
$ WITH ALUMINUM MATERIAL PROPERTIES
$
$GRID:NST,# NODES, X1   , Y1    ,X2    , Y2   ,INC
GRID:  1,    2,    0.0,  0.0,  12.0, &
  0.0,  1  $THIS IS A CONTINUATION OF THE LINE ABOVE -- TWO GRID POINTS
$ HAVE BEEN DEFINED
$
$MEMBER:IMEMB ,N1   ,N2   ,NE    ,MAT
MEMBER: ,1,   2,    1,    ALUMINUM    $ONE MEMBER FROM NODES 1 TO 2 USING
$THE PROPERTIES OF "ALUMINUM" FROM THE MATPROP COMMAND
$
$
$PRINT:DVA INC, COORD,INTF INC,COORD,MM INC,SM INC ?CONDITION LABEL
PRINT:   100,    0,     200,     L,     0,    0  $PRINT DISPLACEMENTS/
$VELOCITIES/ACCELERATIONS EVERY 100 TIME STEPS AND FORCES EVERY 200
         25,     0,     25,      L,     0,    0 ?2 $PRINT MORE OFTEN
$WHEN CONTROL FORCES ARE ACTIVATED
```

Figure 7.4-2 Input Data For Problem 7.4

```
$PLOTINC: PLOT INC  ? CONDITION LABEL
PLOTINC:     10   $STORE RESULTS EVERY 10 TIME STEPS FOR PLOTTING
           3 ?2 $STORE RESULTS MORE OFTEN WHEN CONTROL FORCES ON
$
$BATCH PLOTTING COMMANDS
SNAPSHOT,0.0,4.0,55,0
PLOTC,1,2,3,4,5
PLOTMAG,V,1
PLOTQ,2
PLOTROT,1
$
$INCMASS: NSTEPS,ANGLE?CONDITION LABEL
INCMASS:      1, .0875       $UPDATE MASS MATRIX EACH TIME FOR RIGID ELEMENT
$
$
$ADMASS: NDE1, IDIR1, NDE2, IDIR2, MASS  ?CONDITION LABEL
ADMASS:     1,     1,     1,     1,     @3  $LUMPED MASSES ARE DEFINED AT EACH
            1,     2,     1,     2,     @3  $NODE ACTING IN THE X AND Y DIRECTIONS
            2,     1,     2,     1,     @3  $WITH MAGNITUDE AS GIVEN IN THE DEFINE
            2,     2,     2,     2,     @3  $COMMAND
$
$APPLOAD:LOAD ID,IMEMB ,  DIR , SCALE,TSTART,TSTOP,GRID PNTS ?CONDITION LABEL
APPLOAD:     1,  , Q1,    Q2,   0.0,    4.0,    1        ?2 $APPLY CONTROL FORCES
             1,  , Q3,    Q4,   0.0,    4.0,    2        ?2 $AT NODES 1 AND 2 WHEN
$CL1 IS TRUE,  WITH DIRECTIONS NORMAL TO THE BEAM AND SCALE FACTORS
$PROPORTIONAL TO THE MAGNITUDE OF THE VELOCITY
             1,  ,Q1,    Q2,   0.0,    4.0,    1        ?4
             1,  ,Q3,    Q4,   0.0,    4.0,    2        ?4
             1,  ,Q1,    Q2,   0.0,    4.0,    1        ?5
             1,  ,Q3,    Q4,   0.0,    4.0,    2        ?5
$
$LOAD:LOAD ID, AMPLITUDE
LOAD:    1,       1.0 $DEFINES CONSTANT LOAD WITH AMPLITUDE OF 1.0
$
$
$SET: QN=FORTRAN ?CONDITION LABEL
SET: Q5=10.0*PI/180.0
     Q6=90.0*PI/180.0
     Q7=PI
$
$RESET: QN=FORTRAN ?CONDITION LABEL
RESET: Q1= ANGLE(V,1)  $COMPUTE DIRECTION OF CONTROL FORCE AT NODE 1
       Q2= -MAG(V,1)*1.0E-2  $DEFINE SCALE FACTOR FOR FORCE AT NODE 1
       Q3= ANGLE(V,2)  $COMPUTE DIRECTION OF CONTROL FORCE AT NODE 2
       Q4= -MAG(V,2)*1.0E-2  $DEFINE SCALE FACTOR FOR FORCE AT NODE 2
$
$
$VEL,GRID PT,  DIR, COORD ,INIT VEL
VEL:     1,     3,     0,     -@2      $INITIAL ROTATIONAL AND NORMAL
         2,     3,     0,     -@2      $VELOCITIES ARE DEFINED AT
         1,     2,     0, +18.849556 $GRID POINTS 1 AND 2
         2,     2,     0, -18.849556
```

Figure 7.4-2 continued

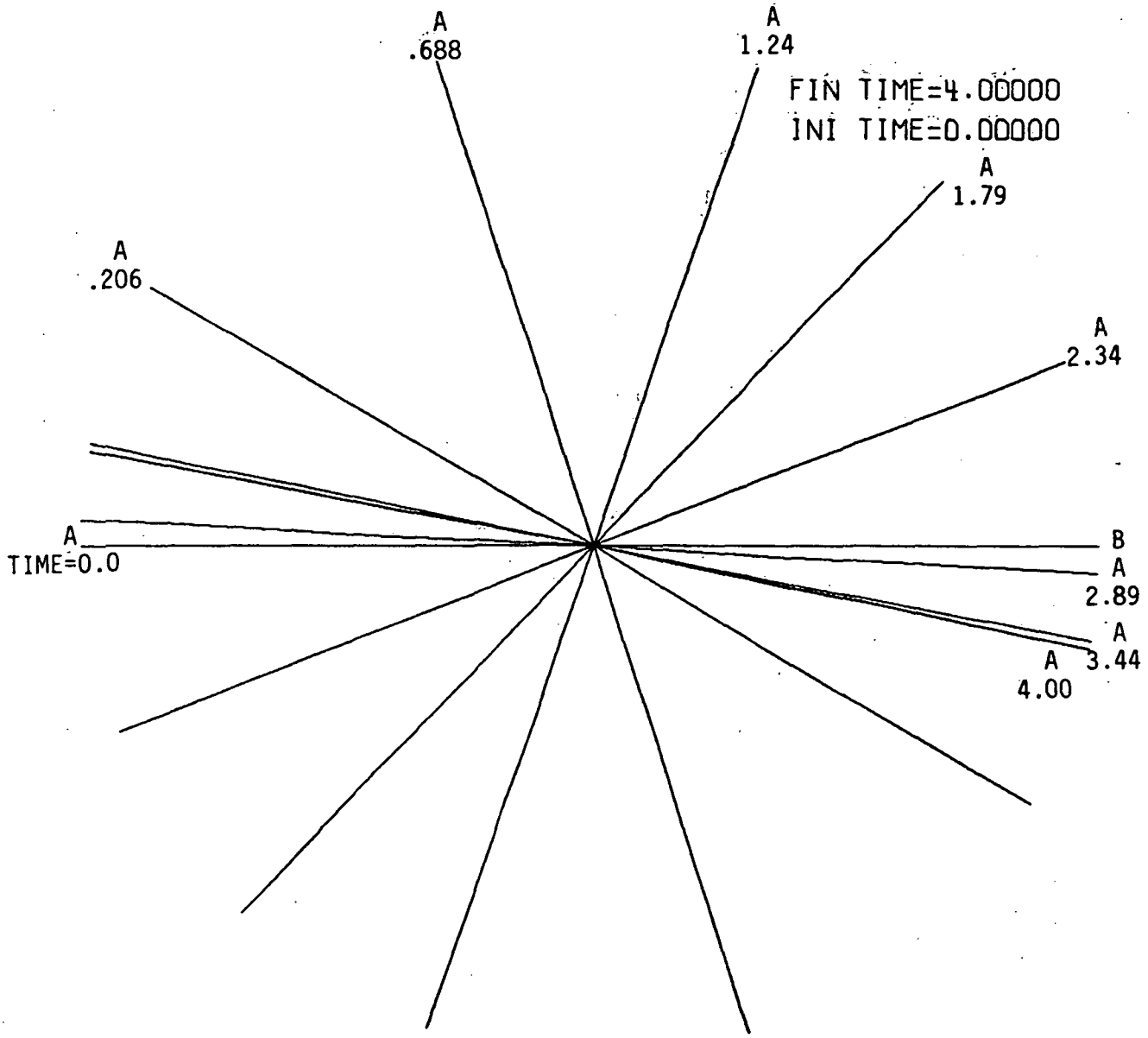


Figure 7.4-3 Plot Created by SNAPSHOT Command

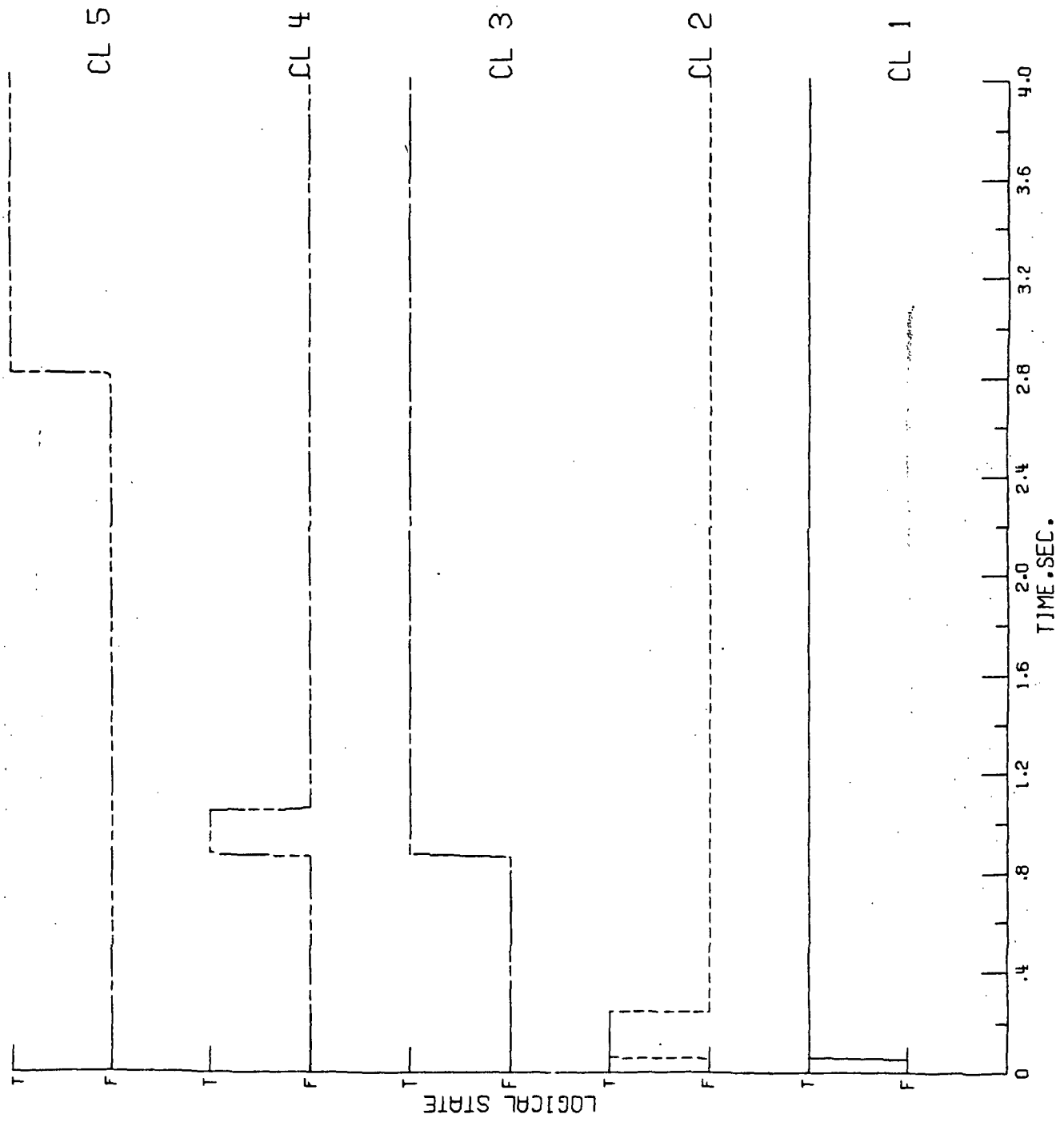


Figure 7.4-4 Plot Created by PLOTC Command

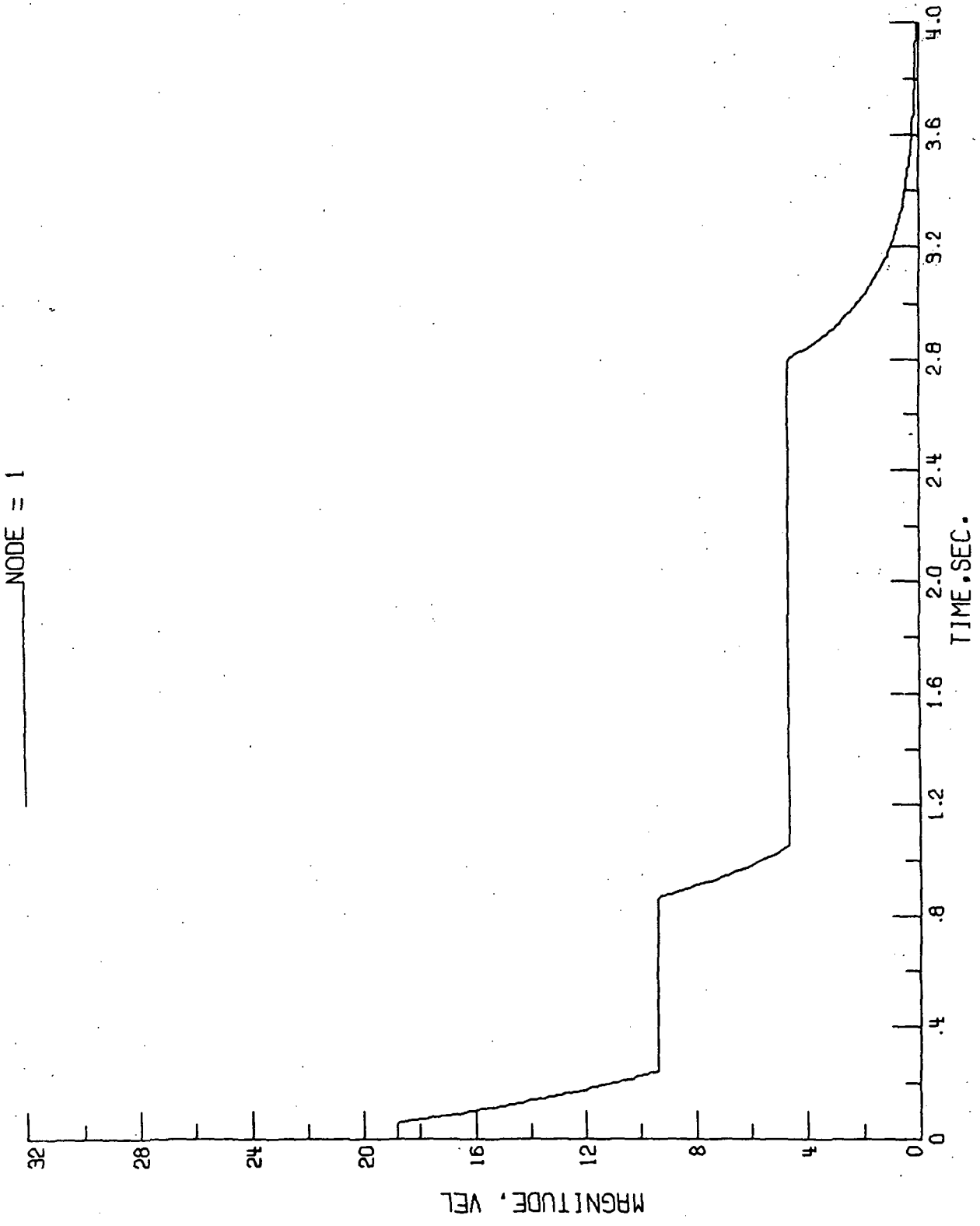


Figure 7.4-5 Plot Created by PLOTMAG Command

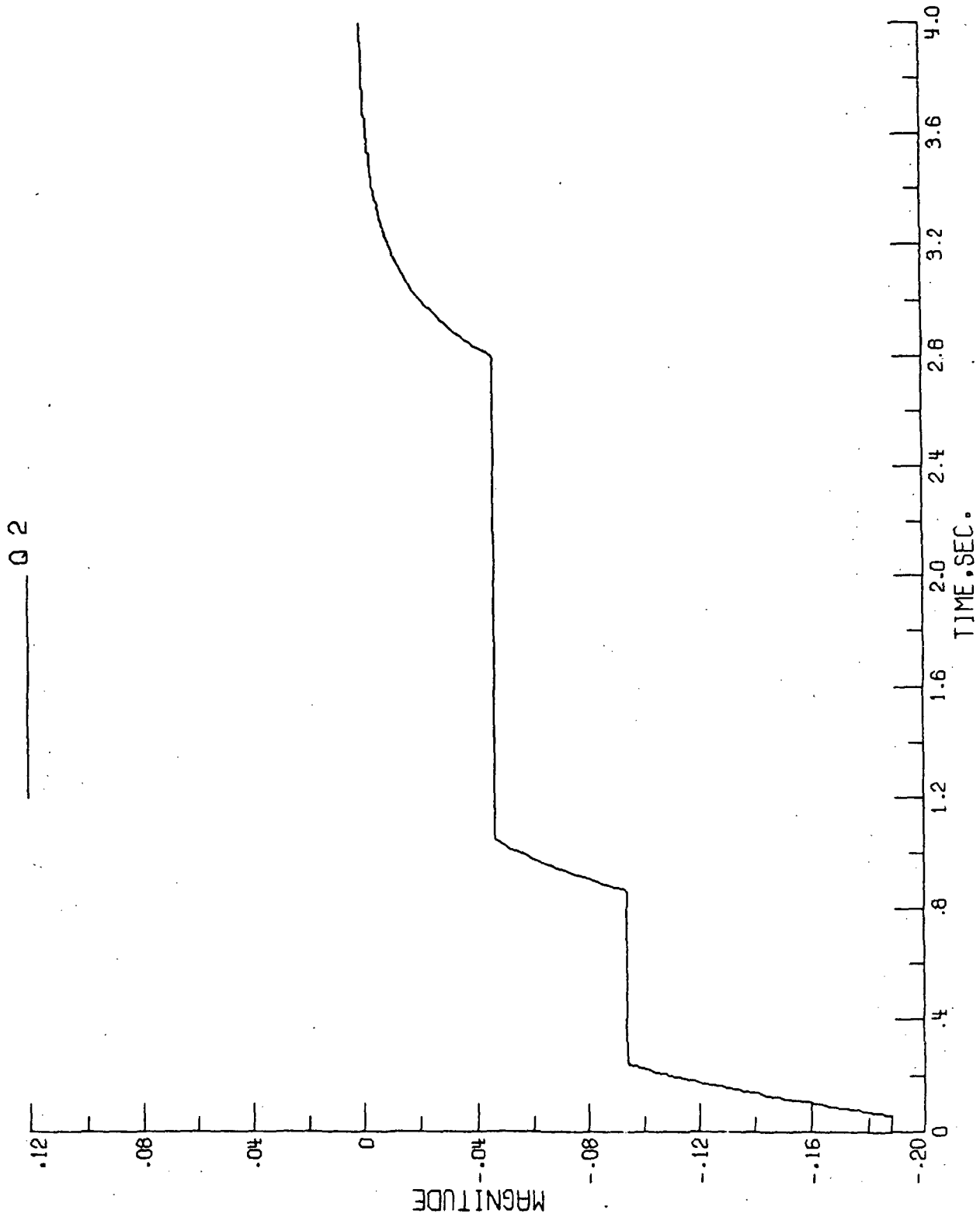


Figure 7.4-6 Plot Created by PLOTQ Command

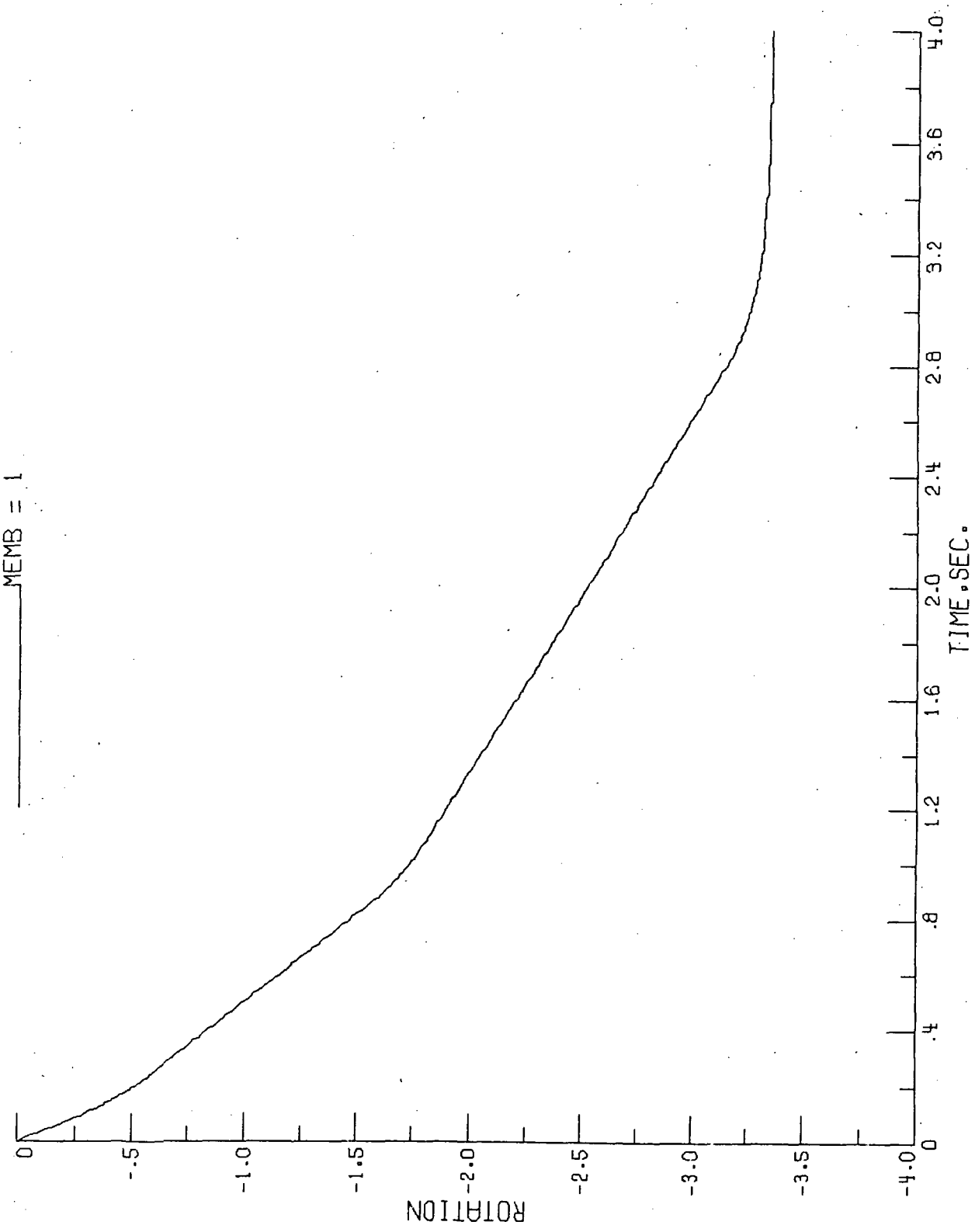


Figure 7.4-7 Plot Created by PLOTROT Command

APPENDIX A: LATDYN STRUCTURE

Components of LATDYN

LATDYN consists of three components which operate in combination with a FORTRAN compiler and are linked through data files and job control language commands. The components are the pre-processor, the LATDYN calculation routines, and the post-processor. The pre-processor acts as the command language interpreter for the program. The post-processor provides the graphics portion of the program.

When the LATDYN program is activated, using the instructions provided in Appendix B for CDC/NOS execution or Appendix C for VAX/VMS execution, all or some of these components are utilized depending on user selected options.

When LATDYN is activated the user selects the components of LATDYN to be executed. For CDC/NOS equipment, the options are selected as arguments of a system control command as shown in Appendix B. For VAX/VMS equipment, the user is queried on the screen concerning option selection.

In addition to the three components, LATDYN also makes use of a FORTRAN compiler. The use of a FORTRAN compiler gives the user access to the power of a high level language, albeit in a highly structured environment. This requires that the FORTRAN compiler be incorporated as part of the execution process of LATDYN and that there be a means of linking the compiled code with the main program. For CDC/NOS and VAX/VMS equipment these requirements are met and utilized automatically.

Pre-processor Component

The principal function of the pre-processor portion of the program is to act as a command language interpreter. It allows the user to modularize the input data by permitting multiple input files. It also serves to check for many syntactical errors which may exist in the input data commands. As described later under this heading, it cannot detect all types of errors, but will detect and flag all command language syntactical errors.

The control instructions which activate the pre-processor depend upon the computer system being used. Appendices B and C provide these instructions for the CDC/NOS and VAX/VMS computers. For the CDC/NOS machines, the pre-processor can be used in either an interactive or batch mode. For the VAX/VMS machine, the pre-processor is used in an interactive mode. Usually the interactive mode will prove most useful.

Pre-processor in the Interactive Mode - After LATDYN has been activated and the pre-processing option in the interactive mode selected, the pre-processor's first action is to ask for the name of the input data file. This query appears on the screen as:

"Where are the input directives?"

The response to this question is a file name or the word "ME". If the response to this question is "ME" then input data commands are to be supplied manually by the user directly from the terminal. The user will be instructed when to supply this data.

The pre-processor creates certain output which may be useful to the user. Pre-processor output is divided into three categories,

* An echo of all input command data
* Warning messages
* Fatal errors

Hence, the next action of the pre-processor is to inquire about the output it creates.

In the interactive mode the program asks for the name of a file for echoing input data:

"Where should the directives be echoed?"

If the response to this question is the word "ME" (or a carriage return), then all pre-processor output (not to be confused with calculated output) is directed to the screen.

If the response is a file name, then input directives will be echoed to that file. When run interactively, the pre-processor always outputs warning and fatal messages to the screen, but in addition the user can place these on files. The program will ask,

" Where should any warning messages be written?"

" Where should any fatal error messages be written ?"

The user should respond to these questions with file names or if only screen messages are desired, with carriage returns. Saving files of these messages can prove useful for large problems wherein many errors may exist.

If the answer to the question "Where are the input directives?" was "ME", a question mark will appear on the screen after the pre-processor output questions have been answered. The user should then enter the individual LATDYN commands following each with a carriage return. The program will respond with an echo of the command if the answer to the question "Where should the directives be echoed?" was "ME". A question mark on the screen indicates that the next command should be entered. When all input data commands have been entered, the user should respond with a single carriage return to terminate manual entry of input data commands.

Whether input data is supplied from a file or from the screen, the user will be queried for additional input data with the question,

"Where are any additional directives?"

The user responds with a file name, "ME" or "NO". This additional request allows input data to be supplied via multiple files or a combination of files and screen input.

Pre-processing in the Batch Mode - This option is only available on the CDC/NOS machines. The instructions for activating the pre-processor in the batch mode are provided on page B-1. Arguments in the instruction command provide the name of an input data file which contains the LATDYN commands.

Files Created by the Pre-processor - When all input data has been supplied, the program will process the data and create the files LATIN, UDAC and POSTIN. The file LATIN contains the pre-processed instructions for the calculation portion of LATDYN. The file UDAC contains any FORTRAN coding supplied by the user in Q-variable definitions and condition statements. UDAC is compiled and linked to LATDYN in the form of subroutine UDAC. In addition, if the input file called for graphics output, a file named POSTIN is created. POSTIN contains all the plotting commands of the input data. However, if a fatal error is detected in the input data, none of these files will not be created.

When the program has completed pre-processing all of the input data and no fatal errors have been detected the program will inform the user that the pre-processor has completed its task.

Note that the existence of these files does not mean that the data is free from all errors. The pre-processor is able to detect certain syntactical errors in the input data. However, there are errors that the pre-processor is unable to detect. The pre-processor cannot detect FORTRAN errors in the Q-variables and the condition statements. Also it cannot check to see that the number of members and grid points used in the model does not exceed that given in the PORDER command. These types of errors will be detected in the FORTAN compiler or in the LATDYN pre-processor. The pre-processor will inform the user of those errors which it can detect.

If a fatal FORTRAN error exists in the condition statements or in the Q-variables then the FORTRAN compiler will detect it only if it is a syntactical error. However, errors such as incorrect or non-existent names of in-line FORTRAN functions are not detectable by the FORTRAN compiler. These will cause execution errors. If such occurs they are caused by errors in the input data. They can be detected by checking the cross reference listing in the compilation listing of Subroutine UDAC and by checking the load map for unsatisfied externals. There should be no undefined variables in UDAC and the loaded LATDYN program should contain no unsatisfied externals for either functions or subroutines. If these errors exist, then the user should correct the spelling or other error in the input data and rerun the pre-processor.

LATDYN Calculation Routines

After successful execution of the pre-processor portion of the program, the calculation routines of LATDYN may be executed. The instructions for doing this depend on the computer being used. Appendices B and C provide these instructions for the CDC/NOS and VAX/VMS computers.

Post-processor Component

The purpose of the post-processor component of LATDYN is to produce graphical output for nearly all quantities evaluated in the calculations portion of the program. When the input data contains a PLOTINC command, LATDYN creates a plot file which contains calculated results at even time intervals as specified in the PLOTINC command. Section 5.18 describes the PLOTINC command. These results may be plotted using the post-processor portion of the program. Commands for the post-processor are described in Section 5.18. These commands determine the actual plots to be created. The post-processor may function in an interactive or batch mode on the CDC/NOS machines, but only interactively on the VAX/VMS machines.

Post-processor in the Batch Mode - On CDC/NOS, the batch post-processing mode is activated using the instructions of Appendix B. Batch post-processing requires a file POSTIN. When post-processor commands are placed in the input data the pre-processor creates POSTIN which contains the post-processor commands of the input data preceeded by the word "batch' and followed by the word "exit". The file may be edited using the system editor and commands added or deleted. Similarly, a POSTIN file can be completely created with the system editor.

Post-processor in the Interactive Mode - Instructions for activating the post-processor in an interactive mode are supplied in Appendices B and C. In the interactive mode, each post-processing command is entered directly from the terminal. During an interactive session, the user has access to two helpful online commands, namely, HELP and INFO. The HELP command allows the user to review the available post-processing commands of section 5.18 with explanation of their functions. The INFO command allows the user to review the data associated with the output plot file which contains the results of the LATDYN execution. This command is useful for it is easy to forget what data is on the output file.

The INFO command provides the the data case title, the initial time, the final time, the number of time steps, the current time window, the number of model grid points, the number of windows and lists of grid point identification numbers, Q-variable numbers and condition numbers.

APPENDIX B: EXECUTION ON CDC CYBER 170 SERIES COMPUTERS UNDER THE NOS OPERATING SYSTEM

This Appendix presents the procedures for executing LATDYN on the CDC/NOS computer system operating at the NASA Langley Research Center. Procedures for other computer installations will be similar.

LATDYN consists of three components: a pre-processor, LATDYN calculation routines and a post-processor. The user has a number of options available to execute the various components of LATDYN using procedure files existing on the LATDYN library (UN=LATDYN) at LaRC. These include:

1) execution of pre-processor and LATDYN calculation routines
2) execution of post-processor
3) execution of all three components

For any of these options, execution is carried out via procedure files. At LaRC these files are obtained from UN=LATDYN using the following,

```
GET,"PROC FILE NAME"/UN=LATDYN
```

Also note that most of the procedure files require the user to have a file named "ZLATDYN" on his user number which contains the following information,

```
USER,charge number,password.
CHARGE,xxxxxx,LRC.
DELIVER delivery information
```

1) Execution of Pre-Processor and LATDYN Calculation Routines:

a) The user can execute LATDYN in a batch mode, an interactive mode or a combination of each. The pre-processor may be executed interactively to enable the user to carry out a syntactical check on the data set. To execute the pre-processor interactively and the calculation routines in either batch or interactive modes the procedure files "LATRUNB" or "LATRUNI" are used. The commands take the form,

```
LATRUNB,
   or   DATA=_____,PLOT=_____,MACH=_____,OUT=_____
LATRUNI,
```

where,

DATA=name of the input data file
PLOT=name of the output data file for post-processing (default=TAPE12)
MACH=computer specification for batch execution (default=RHC)
OUT=name of output file for printed results (default=LATOUT)

Following the successful execution of the pre-processor in an interactive mode, the LATDYN calculation routines are executed in the batch mode if "LATRUNB" was used or the interactive mode if "LATRUNI" was used. The output data file for post-processing produced by either a "LATRUNB" or "LATRUNI" procedure file will be in indirect access form.

b) A batch job can also be submitted from the terminal without executing the pre-processor interactively. Thus all syntactical checks are performed in the batch mode, and the user will not immediately know if the job is able to be executed.

In this case, the procedure files "LATBTCH" or "LATDIR" are used. "LATBTCH" is used if the output data file for post-processing is to be indirect access and "LATDIR" if the output data file for post-processing is to be direct access. The command takes the form,

```
LATBTCH,
  or     DATA=_____,PLOT=_____,MACH=_____,OUT=_____
LATDIR,
```

2) Execution of Post-processor :

a) Interactive post-processing can be done at any time following the execution of LATDYN calculation routines provided the output data file for post-processing has been saved. Following the interactive session the user has the option to get hard copies of all the plots made by following the instructions which appear on the screen.

For interactive post-processing, use procedure file "LATPLOT" when output data has been saved in indirect access form and use "LATPLTD" when output data has been saved in direct access form. Commands take the form,

```
LATPLOT,PLOT=_____
    or
LATPLTD,PLOT=_____
```

Following the interactive plotting session the user will automatically receive instructions for obtaining hard copies of the plots via the procedure file LATPOST. LATPOST contains the commands for converting the plot file, created by LATPLOT or LATPLTD, into a SAVPLT file necessary for the creation of hard copies. The command for obtaining hard copies is,

```
LATPOST,HPLOT=_____,DEVICE=_____
```

where,

HPLOT=name of the savplt file to store the hard copy plotting information (default=SAVPLT)
DEVICE=type of hard copy device to create hard copies (currently limited to VARIAN)

b) The user can also perform post-processing in the batch mode by using the procedure file "PLTBTCH". This file requires, in addition to the output data file for post-processing, that the user have a command file which contains the plotting instructions for executing the post-processor. The form of the command is,

```
PLTBTCH,PLOTDAT=_____,DATA=_____,DEV=_____,TYPE=_____,
        MACH=_____,PLTSV=_____
```

where,

PLOTDAT=name of the LATDYN output data file for post-processing (default=TAPE12)
DATA=name of the file containing the plotting instructions (default=POSTIN)
DEV=type of hard copy device to be used and can take the form,

VAR (Varian plotter), (default)
VARF (Varian fanfold)
CAL11 (Calcomp)
CALCM (Calcomp 33 inch flatbed)
MOVIE (Movies)

TYPE=output plot file specification
TYPE=IND if the file "PLOTDAT" is indirect access (default)
TYPE=DIR if the file "PLOTDAT" is direct access
MACH=computer specification (default=RHD)
PLTSV=name of the savplt file which the hard copy plotting information is to be stored under (default=SVPLT).

3) Execution of All Three Components:

The user can also submit a single batch job which will execute the pre-processor, the LATDYN calculation routines and the post-processor. The plots generated by the execution of the post-processor are in accordance with the plotting instructions included in the LATDYN data file as described in Section 5.19.

To execute in this fashion use procedure file LATDYNA in the following manner,

```
LATDYNA,DATA=_____,PLOT=_____,POSTDAT=_____,TYPE=_____,
        DEV=_____,MACH=_____,OUT=_____,PLTSV=____
```

where,

DATA=name of input data file
PLOT=name of output data file for post-processing
(default=TAPE12)
POSTDAT=name of file containing the plotting instructions
(default=POSTIN)
TYPE, DEV, MACH, OUT and PLTSV are as defined above

Note: This execution option is not currently available.

APPENDIX C: EXECUTION ON DEC VAX COMPUTER UNDER THE VAX/VMS OPERATING SYSTEM

The VAX computer is a 32 bit virtual memory machine and is well able to handle the computational requirements for LATDYN. Though LATDYN was originally written for the 60-bit CYBER 6000 series of computers, it can produce good results using the 32-bit arithmetic of VAX's single precision mode. However, only the double precision version of the program should be used if independent verification of results are unavailable.

There are two files for executing LATDYN on the VAX system. The files are 'LATINT.COM' for the interactive execution of the program and 'LATSUB.COM' for the batch execution of the program. Because of the long execution time required for the double precision version of LATDYN, it is recommended that the user use the batch mode unless the run is very short. Both files are used interactively and require user response to queries.

The VAX VMS operating system uses the @ symbol to initiate the execution of a command file. File LATINT.COM and LATSUB.COM are executed by typing '@LATINT' or '@LATSUB'. The '.com' is not included in the command.

Interactive Execution

To execute LATDYN interactively on the VAX/VMS system, the instruction "@LATINT" is used. When entered, the program will request the names of the data files to be used and created. The files to be named are the pre-processor echo file, restart input data file, restart output data file, plot output data file and print output data file. The pre-processor echo file is not supplied by the user, but is used only to echo the input data file onto the print output data file. The restart input file is required for a restart and is generated by a previous LATDYN execution. The default name is FOR010.DAT. The restart output data file must be saved if the user wants the to perform a restart in the future. The default name is FOR010.DAT. The plot output file is created if the PLOTINC command is present in the input data. The default name is FOR012.DAT. The print output file name must be given if the user does not wish to use the default name of LATOUT.DAT.

Batch Execution

To execute the calculation routines of LATDYN in the batch mode with the pre-processor component of LATDYN executed in the interactive mode, the instruction "@LATSUB" is used. After successful pre-processing, a submittal job file is automatically created and submitted for batch processing. The use of LATSUB is recommended for all but very small jobs.

After entering the instruction "@LATSUB" the program will request a job root number. The root number will allow the user to submit simultaneous cases. Next, requests will be made for the

names of files to be used or created just as is done in the case of interactive execution.

If the pre-processor executes successfully, the submitted job will be placed in the system queue for execution. The user is able to check the status of the job by using the VAX DCL command "SHOW". The submittal command uses the VAX DCL command "NOTIFY" to inform the user when the program has completed execution. The user is able to logoff the system or perform another task if so desired.

REFERENCE: LATDYN THEORY

Following is the reference document which describes the theoretical basis for the LATDYN computer program.

CONVECTED TRANSIENT ANALYSIS FOR LARGE SPACE STRUCTURES MANEUVER AND DEPLOYMENT

Jerrold M. Housner
NASA Langley Research Center
Hampton Virginia 23665

Presented at the AIAA/ASME/ASCE/AHS 25th Structures, Structural Dynamics and Materials Conference

AIAA Paper No. 84-1023-CP

Palm Springs, California

CONVECTED TRANSIENT ANALYSIS FOR LARGE SPACE STRUCTURES MANEUVER AND DEPLOYMENT

Jerrold Housner *
NASA Langley Research Center
Hampton Virginia 23665

Abstract

An application of convected finite element transient analysis to maneuver and deployment of flexible multi-member trusses and frames for application to large space structures is presented. The use of convected analysis within the finite element framework permits accurate treatment of both large elastic deformations and large rigid body rotations, while permiting the utilization of a vast storehouse of existing computional finite element technology. A consistent mass (rather than a lumped mass) formulation of the equations of motion is developed so that exact rotational inertial properties are retained in the analysis. This is critical for space structures which undergo unlimited rotations. A shortcoming of consistent mass formulations for rotating elastic systems is that they lead to inertial terms which depend upon elastic motion and thus require up-dating. It is shown that this shortcoming may be avoided by making appropriate approximations with little loss in accuracy. The analysis is used to study maneuver of slender booms, deployment of axisymmetric hoops composed of flexible members and deployment of planar unfolding multi-flexible-member structures. It is found that multi-member structures can possess natural deployment patterns such as a sequential (member-by-member) pattern. Such patterns may be highly desirable features of deployment design and their achievement without the use of complex control mechanisms could represent considerable cost savings.

Nomenclature

$a_{i,1}, a_{1,2}$	coefficients of i^{th} beam element flexural shape function
$A_{i,11}$, $A_{i,12}$ $A_{i,21}$, $A_{i,22}$	integrals of flexural shape function as given in Appendix B
[B]	connectivity matrix, eq. (20)
[C]	constraint matrix, eq. (22)
[D]	damping matrix
$(\hat{d}_i)$	vector of displacement degrees-of-freedom for i^{th} finite element
(d)	full vector of displacement degrees-of-freedom, see eq. (20)
$(\hat{f}_i)$	vector of generalized forces for i^{th} finite element, see Appendix A
(f)	full vector of generalized forces
(F)	vector of external forces
$(\hat{G}_i)$	vector of nonlinear inertial terms for the i^{th} finite element.
(G)	total vector of nonlinear inertial terms
$(\hat{G}_i^{(j)})$	components of $(\hat{G}_i)$; $(\hat{G}_i) = \sum_{j=1}^{4} (\hat{G}_i^{(j)})$
$g_i^{(j)}{}_{,mn}$	entries of $(\hat{G}_i^{(j)})$ as given in Appendix B; $1 \leq m \leq 6$; $1 \leq n \leq 6$
h_i	bending displacement of i^{th} finite element beyond the i^{th} convected coordinate, eq. (13)
$H_{i,j}$	integrals of the flexural shape function; $1 \leq i \leq N$; $1 \leq j \leq 5$, see Appendix B
k	rotational spring constant at joints of deployable structures
$[K_s]$	spring constant matrix, see eq. (30)
L_i	length of i^{th} finite element
m_i	mass of i^{th} finite element
$m_i^{(j)}{}_{,rs}$	entries of $[\hat{M}_i^{(J)}]$; $1 \leq r \leq N$; $1 \leq s \leq N$, see Appendix A
[M]	assembled mass matrix
$[\hat{M}_i]$	mass matrix for i^{th} finite element
$[\hat{M}_i^{(j)}]$	components of mass matrix; $[\hat{M}_i] = \sum_{j=1}^{3} [\hat{M}_i^{(j)}]$
N	total number of members
N_D	total number of independent degrees of freedom
r, R	local radius of hoop member during deployment at its center and either end, respectively
R_h	fully deployed hoop radius
$\bar{s}$	local dimensionless coordinate along convected beam finite element; $0 \leq s \leq 1$

* Aerospace Engineer, Member AIAA

$[T_i]$	convected coordinate transformation matrix for i^{th} finite element
u, v	local displacement of beam element parallel to x and y axis respectively
x, y	global cartesian coordinate system
α_i	angle formed by i^{th} undeformed finite element and x axis
$\Gamma_{i,11}$, $\Gamma_{i,12}$, $\Gamma_{i,21}$, $\Gamma_{i,22}$	integrals of flexural shape functions, see Appendix A
δ	variational operator
$\epsilon_{m,i}$	mean axial strain of straight line joining end points of i^{th} finite element
$\epsilon_{o,i}$	mean axial strain of i^{th} deformed finite element
$\eta_{i,1}$, $\eta_{i,2}$	end point displacements of i^{th} finite element parallel to x axis
θ_i	angle formed by straight line joining the end points of i^{th} finite element and the x axis
$\xi_{i,1}$, $\xi_{i,2}$	end point displacements of i^{th} finite element parallel to y axis
$\phi_{i,1}$, $\phi_{i,2}$	rotations of i^{th} finite element end points
ψ	angle formed by hoop member and vertical axis, see figure 7

Introduction

Recently, interest has grown in establishing permanently manned space stations. Space stations may be composed of multi-body interconnected structural components some of which are thin shells (habitation and work modules), panels (solar arrays and radiators) or beams (space booms, towers and robotic manipulator arms). These components or the entire space station may, from time to time, be maneuvered or, as in the case of robotic arms, manipulated through large rotations. Such maneuvers result in dynamic loads on the various space station components. It is likely that to obtain structural component loads for preliminary design, maneuvers of large beam-frame models will be analyzed.

In addition, many station components may be sent aloft packaged and then deployed in orbit. Deployment concepts often involve large rotations of flexible truss-type members. Since deployment experiments are both difficult and expensive to perform either on the ground or in orbit, dynamic analyses will be relied upon to validate performance of proposed deployment concepts.

In any of the three application areas, maneuver, manipulation or deployment, dynamics of interconnected flexible components undergoing large rotations is involved. In some cases involving flexibility and large angle rotation, flexible and rotational responses are uncoupled and a linear solution is possible.

This situation can occur when the following conditions are satisfied:

(i) the structural modes (including rigid body modes) do not change with time

(ii) the contributions of the external forces in each modal shape (including rigid shape) do not depend on the structure's rotated position

(iii) the structural deformations are small

(iv) the nonlinear centrifugal and Coriolis forces are negligible.

When only some of these conditions are met, it may still be accurate to perform a quasi-linear analysis in which the modes and/or external force contributions in the modal shapes are updated periodically in the course of the analysis. However, there exists a large class of problems in which the nonlinear coupling between flexibility and large angle rotation must be accounted for.

In space structures problems requiring a nonlinear analysis, the strains in the structural components are small even though rotations and hence displacements are large. To avoid the computations associated with accurate strain measures for large motions, it is desirable to separate the rigid body rotations from the structural deformations. This separation may be accomplished by using an updated or convected coordinate system.

Over the past decade, convected coordinates have been succesfully used in conjunction with finite element models in performing transient response analyses using explicit time integration algorithms. (For example, see references 1-4). Thus, convected transient analysis techniques offer a viable approach for analyzing maneuver, manipulation and deployment. Two additional benefits to this approach are the utilization of a vast storehouse of finite element computational technology and the capability of the convected analysis within the finite element framework to account for both large elastic deformations and large rigid body rotations.

Even though convected transient analysis has seen considerable use, its application to fundamental configurations which are likely to arise in space stations and their components has not received much attention. Previous developments in convected frame analysis have been limited to structures containing large concentrated masses, such as automobiles where the engine, transmission, differential, etc, may be treated as lumped masses. A lumped mass formulation has considerable numerical advantages over a consistent mass formulation. However, many flexible components of large space systems do not lend themselves to an accurate lumped mass treatment. (See reference 5.) For example, large space booms and antenna structures have uniform or slowly varying distributed mass. Lumping the distributed mass leads to overestimating the rotational inertia and thus underestimating

its rigid body rotational rate. In addition, the coupling between rigid body motion and structural deformation due to nonlinear centrifugal and Coriolis forces or large structural deformations may be inaccurately represented in a lumped mass formulation. Thus a consistent mass formulation is desirable.

The purpose of this paper is to present the development and application of a consistent mass finite element formulation for convected transient analysis of two-dimensional beam-frame configurations for use in space. The computational consequences of a consistent mass formulation are discussed with emphasis on reducing computations through appropriate approximations. The analysis is applied to rotational maneuver and translation of long slender booms, deployment of an unfolding multi-flexiblemember axisymmetric hoop and deployment of a planar unfolding multi-flexible-member structure.

Convected Analysis

The purpose in using a convected analysis is to separate rigid body rotations from structural deformations for in each finite element. The need for this separation arises since any truncated form of the physical strain measure in a fixed coordinate system is inaccurate for large rotations. Separation may be accomplished by setting up a collection of convected coordinate systems which follow the rotation of each finite element. The convected analysis used herein deviates from that of references 1 thru 3 in that the development of the inertial terms uses a consistent mass formulation rather than a lumped mass formulation. The consistent formulation is important when rigid body rotations must be accurately accounted for. To illustrate this consider the rotation of a single rigid beam member. Figure 1 illustrates the error from the exact rotational inertia associated with a lumped mass or diagonal mass matrix formulation. This simple illustration demonstrates that when a lumped mass formulation is used, five rigid finite elements are required to approximate the exact rotational inertia to within eight percent. On the other hand, a consistent mass formulation requires only one rigid finite element for the exact value.

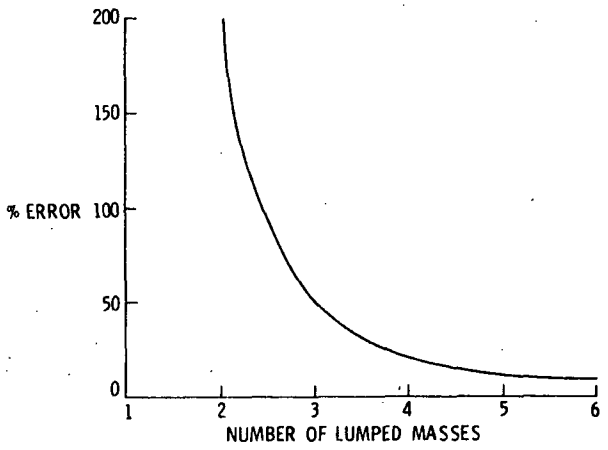


Figure 1. Percent Error in Centroidal Rotational Inertia For Lumped Mass Modeling of Beam Member [Error=(Approx. value)/(Exact value) - 1]

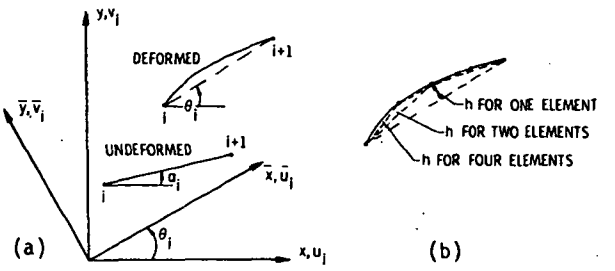


Figure 2. Finite Element Deformation and Convected Coordinate System
(a) Rotation of Convected Axes
(b) One, Two and Four Finite Element Modeling of a Single Beam Member

Coordinate system

Consider the deformed beam finite element shown in figure 2a. The convected motion of the element is defined by the straight line which joins the ends of the element. Deviation from this lines constitutes the deformation of the element. A convected coordinate system, local to each element is thus defined which rotates with the element, but does not translate with it. (The convected coordinate system could be defined to translate with the element, but this is not necessary.) The convected system in general differs for each element. Coordinates in the global stationary coordinate system are related to those in the i^{th} convected system by

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = [T_i] \begin{pmatrix} \bar{x}_i \\ \bar{y}_i \end{pmatrix} \tag{1}$$

where barred and unbarred quantities denote measurements in the convected and stationary coordinate systems respectively and

$$[T_i] = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix} \tag{2}$$

The convection angle θ_i for the i^{th} element is defined by

$$\tan\theta_i = (L_{i,y} + \Delta\xi_i)/(L_{i,x} + \Delta\eta_i) \tag{3}$$

where L_{ix} and L_{iy} are the projected lengths of the i^{th} undeformed element on the x and y axes respectively and

$$\Delta\eta_i = \eta_{i,2} - \eta_{i,1}$$

$$\Delta\xi_i = \xi_{i,1} - \xi_{i,2}$$

in which $\eta_{i,1}$, $\eta_{i,2}$, $\xi_{i,1}$ and $\xi_{i,2}$, are the values of grid point displacements at the ends of the i^{th} element and parallel to the x and y axes as shown in figure 2a.

The displacements in the convected system at any point on the i^{th} element transform as

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = [T_i] \begin{pmatrix} \bar{u}_i \\ \bar{v}_i \end{pmatrix} \tag{4}$$

However, the velocities transform as

$$\begin{pmatrix}\dot{u}_i\\ \dot{v}_i\end{pmatrix} = [T_i]\begin{pmatrix}\dot{\bar{u}}_i\\ \dot{\bar{v}}_i\end{pmatrix} + [T_i][I^*]\begin{pmatrix}\bar{u}_i\\ \bar{v}_i\end{pmatrix}\dot{\theta}_i \qquad (5a)$$

where,

$$[I^*] = \begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}$$

and the accelerations transform as,

$$\begin{pmatrix}\ddot{u}_i\\ \ddot{v}_i\end{pmatrix} = [T_i]\begin{pmatrix}\ddot{\bar{u}}_i\\ \ddot{\bar{v}}_i\end{pmatrix} - 2[I^*]\begin{pmatrix}\dot{\bar{u}}_i\\ \dot{\bar{v}}_i\end{pmatrix}\dot{\theta}_i - [I^*\ddot{\theta}_i + I(\dot{\theta}_i)^2]\begin{pmatrix}\bar{u}_i\\ \bar{v}_i\end{pmatrix} \qquad (5b)$$

In equation (5b) the second and fourth terms on the right hand side are recognized as the Coriolis and centrifugal contributions, respectively, to the absolute acceleration vector. Converse relationships for equations (5a) and (5b) may readily be derived.

Rotation and Strain Rates

The rotational rate of the straight line joining the end points of the i^{th} finite element may be found by differentiating eq. (3) with respect to time and the mean axial strain rate may be found by differentiating with respect to time the relation

$$\varepsilon_{i,m} = \sqrt{(L_{i,x} + \Delta n_i)^2 + (L_{i,y} + \Delta\xi_i)^2} - 1 \qquad (6)$$

The result is

$$\begin{pmatrix}\dot{\varepsilon}_{i,m}\\ (1+\varepsilon_{i,m})\dot{\theta}_i\end{pmatrix} = (1/L_i)[T_i]^T\begin{pmatrix}\Delta\dot{n}_i\\ \Delta\dot{\xi}_i\end{pmatrix} \qquad (7)$$

For small strains, $\varepsilon_{i,m}$ on the left hand side of eq. (7) may be neglected compared to unity.

Finite Element Displacements

The finite element displacements may be approximated in the convected system as,

$$u_i(\bar{s}) = \eta_{i,1} + \bar{s}\Delta n_i \qquad (8a)$$

$$v_i(\bar{s}) = \xi_{i,1} + \bar{s}\Delta\xi_i + h_i(\bar{s}) \qquad (8b)$$

where, h_i is the flexural displacement and

$$h_i(0) = h_i(1) = 0$$

and

$$0 \le \bar{s} \le 1$$

The local strain at the neutral axis of the i^{th} element may be found from by adding on to $\varepsilon_{i,m}$, the usual first-order neutral axis strain due to flexure to yield,

$$\varepsilon_{i,o} = \varepsilon_{i,m} + (1/2L_i^2)(h_i')^2 \qquad (9)$$

and the axial strain at any point through the thickness of the element is,

$$\varepsilon_i = \varepsilon_{o,i} - (z/L_i^2)v_i''$$

where z is a thickness coordinate measured from the neutral axis.

Virtual Work

The virtual work contribution from the internal forces of the i^{th} element is given by,

$$\delta U_i = L_i\int_0^1 (P_i\delta\varepsilon_{i,o} - (M_i/L_i^2\delta v_i'']d\bar{s} \qquad (10)$$

where P_i is the average axial load over the length of the i_{th} element and M is the bending moment in the i^{th} element given by,

$$M_i = -(EI)_i\, v_i''/L_i^2$$

From eqs. (7) and (9),

$$\delta\varepsilon_{i,o} = (\delta\Delta n_i\cos\theta_i + \delta\Delta\xi_i\sin\theta_i)/L_i + h_i'\delta h_i'/L_i^2 \qquad (11)$$

Substituting eqs.(11) and (8b) into eq. (10) gives,

$$\delta U_i = P_i(\delta\Delta n_i\cos\theta_i + \delta\Delta\xi_i\sin\theta_i) + (1/L_i^2)\int_0^1 M_i\delta h_i'd\bar{s} + (1/L_i)\int_0^1 P_i h_i'\delta h_i'd\bar{s} \qquad (12)$$

Herein the flexural displacement is defined as,

$$h_i(s) = a_{i,1}(s)\phi_{i,1} + a_{i,2}(s)\phi_{i,2} - [a_{i,1}(s) + a_{i,2}(s)](\theta_i - \alpha_i) \qquad (13)$$

where $\phi_{i,1}$ and $\phi_{i,2}$ are rotations of nodes 1 and 2 at the ends of the i^{th} element. Substituting into eq. (12) yields,

$$\delta U_i = (\hat{f}_i)(\delta\hat{d}_i) \qquad (14)$$

where

$$(\hat{d}_i) = (\eta_{i,1}, \xi_{i,1}, \phi_{i,1}, \eta_{i,2}\ \xi_{i,2}, \phi_{i,2})^T$$

and the expressions for entries of the internal force vector (f_i) are given in Appendix A.

For a consistent mass formulation, the contribution of the i th element to the total virtual work done by the inertial forces is expressed as,

$$\delta W_i = m_i\int_0^1 (\delta u_i,\ \delta v_i)(U_i, V_i)^T d\bar{s} \qquad (15)$$

where m_i is the uniform mass of the i^{th} element and the expressions for δu_i and δv_i may be found by substituting the variations of eqs. (8a) and (8b) into eq. (5) to yield

$$\begin{pmatrix}\delta u_i\\ \delta v_i\end{pmatrix} = \begin{pmatrix}\delta\eta_{i,1} + \bar{s}\delta\Delta n_i\\ \delta\xi_{i,1} + \bar{s}\delta\Delta\xi_i\end{pmatrix} + \begin{pmatrix}-h_i\delta\theta_i\cos\theta_i - \delta h_i\sin\theta_i\\ -h_i\delta\theta_i\sin\theta_i + \delta h_i\cos\theta_i\end{pmatrix} \qquad (16a)$$

The expressions for $\ddot{u}_i$ and $\ddot{v}_i$ are found by differentiating eqs. (8a) and (8b) and substituting into eq.(6) to yield,

$$\begin{pmatrix}\ddot{u}_i\\ \ddot{v}_i\end{pmatrix} = \begin{pmatrix}\ddot{\eta}_{i,1} + s\Delta\ddot{n}_i\\ \ddot{\xi}_{i,1} + s\Delta\ddot{\xi}_i\end{pmatrix} + \begin{pmatrix}-\ddot{h}_i\sin\theta_i - 2\dot{\theta}_i\dot{h}_i\cos\theta_i - h_i\ddot{\theta}_i^2\sin\theta_i - \dot{\theta}_i\dot{h}_i\cos\theta_i\\ \ddot{h}_i\cos\theta_i - 2\dot{\theta}_i\dot{h}_i\sin\theta_i - h_i\ddot{\theta}_i^2\cos\theta_i - \dot{\theta}_i\dot{h}_i\sin\theta_i\end{pmatrix} \qquad (16b)$$

In each of equations (16a) and (16b) the first column vector on the right hand side arises from the rigid body motion of the finite element while the second term arises from the flexural deformations of the element. Coriolis and centrifugal terms are explicit in the the second term of eq. (16b) and implicit in the first term. The implicit effect may be observed by transforming the first term of eq. (16b) to convected coordinates via eq. (6) and then evaluating the result at the element end points.

When the product of eqs. (16a) and (16b) is taken to produce the integrand of eq. (18) and the indicated integration is performed, rigid body rotation and flexural deformation terms are coupled. The result for the i^{th} element may be expressed as

$$\delta W_i = m_i(\delta\hat{d}_i)[\hat{M}_i](\ddot{\hat{d}}_i) + m_i(\delta\hat{d}_i)^T(\hat{G}_i) \qquad (17)$$

where $[\hat{M}_i]$ is the mass matrix of the i^{th} finite element, and $(\hat{G}_i)$ is a vector of nonlinear kinematic terms.

The mass matrix contains terms due to rigid body motion and the predominant inertial coupling between rigid body motion and flexural deformation. The treatment of the mass matrix is simplified by observing that it may be separated into three distinct parts, namely,

$$[\hat{M}_i] = \sum_{j=1}^{3}[\hat{M}_i(j)]$$

The entries of $[\hat{M}_i(j)]$ are given in Appendix B. Entries of $[\hat{M}_i(1)]$ are independent of i and arise from rigid body motions and stretching of the i^{th} finite element while those of $[\hat{M}_i(2)]$ and $[\hat{M}_i(3)]$ arise from flexural motions of the i^{th} finite element. The entries of both $[\hat{M}_i(2)]$ and $[\hat{M}_i(3)]$ depend on the convected rotation θ_i and hence would generally require up-dating during the analysis. Furthermore, those entries requiring up-dating contain h_i.

As the size of the finite element is reduced, the value of h_i decreases as is noted in figure 2b and then the entries of $[\hat{M}_i(3)]$ also decrease. Thus for a sufficiently fine finite element grid, the contribution of $[\hat{M}_i(3)]$ to the total virtual work becomes negligible and if retained would require few up-dates during the analysis. Furthermore, if an element is rigid, then from eq. (13),

$$\phi_{i,1} = \phi_{i,2} = \theta_i \qquad (18)$$

It may then be shown that the contributions of $(\delta d_i)[\hat{M}_i(2)](\ddot{\hat{d}}_i)$ to the virtual work of the inertial forces vanishes for a rigid element and furthermore for a stiff element, approaches zero as h approaches zero. It follows, that for relatively stiff elements, few up-dates of $[\hat{M}_i(2)]$ will be required. For the cases considered herein, it was found that even for booms with slenderness ratios up to 1000, $[\hat{M}_i(2)]$ still required few up-dates and $[\hat{M}_i(3)]$ could be entirely neglected when two finite elements were used per half-wave of beam member deformation.

Similarly the vector $(\hat{G}_i)$ may be separated as,

$$(\hat{G}_i) = \sum_{j=1}^{4}(\hat{G}_i(j))$$

whose entries are given in Appendix B. These vectors arise from coupling between element flexure and nonlinear centrifugal and Coriolis effects. All five vectors vanish as h_i approaches zero. Furthermore, for small h_i, $(G_i(1))$, $(G_i(2))$ and $(G_i(3))$ behave as h_i, while $(G_i(4))$ behaves as h_i^2. (This does not imply that all centrifugal and Coriolis effects vanish as h_i approaches zero, for these effects are implicit in the piecewise representation of the entire member).

As a consequence of the foregoing discussion, it is clear that many terms in the governing equations will be quite small in many applications and either be negligible or require few up-dates.

Equations of Motion and Constraint

The equations of motion may be derived from the principle of virtual work using eqs. (14) and (17)

$$\sum_{i=1}^{N}(\delta\hat{d}_i)^T\{m_i[\hat{M}_i](\ddot{\hat{d}}_i)+m_i(\hat{G}_i)+(\hat{f}_i)\} = \delta We \qquad (19)$$

where δWe is the virtual work of applied external forces and N is the total number of beam members. Lumped masses are accomodated by appropriate additions to the diagonal entries of $[\hat{M}_i]$. Equation (19) yields the equations of motion if the entries of $(\delta\hat{d}_i)$ are independent. However, they are not independent since members are connected together and since multi-point constraints or other constraints may exist. An independent set of displacements can be formed by utilizing the constraints on the system.

Connectivity Constraints

Connectivity relations may be expressed by the matrix equation,

$$((\hat{d}_1)(\hat{d}_2)_j,\ldots,(\hat{d}_N))^T = [B](d) \qquad (20)$$

where (d) is a vector of N_D degrees of freedom. The connectivity matrix [B] will usually contain many zeros and ones. Rather than substitute eq. (20) into eq. (19), connectivity is usually accounted for in finite element computer implementation when the mass matrix and internal force vectors are assembled. This is accomplished by appropriate superposition of entries in $(\hat{G}_i)$, $(\hat{f}_i)$ and $[\hat{M}_i]$ as directed by the connectivity array and computer storage addresses. Thus after assembly, eq. (19) effectively takes on the form,

$$(\delta d)^T\{[M](d) + (G) + (f)\} = (\delta d)^T(F) \qquad (21)$$

where (F) is a vector of external forces.

Linear Multi-point Constraints

For N_C (less than N_D) multi-point constraints one has,

$$[C_{DD} \vdots C_{DI}]\begin{pmatrix} d_D \\ -- \\ d_I \end{pmatrix} = 0$$

where the constraint matrix and displacement vector (d) have been partitioned into dependent and independent sets. The selected set of dependent degrees-of-freedom be expressed in terms of the independent set of degrees-of-freedom as

$$(d_D) = [C](d_I)$$

where,

$$[C] = -[C_{DD}]^{-1}[C_{DI}] \tag{22}$$

Eq. (22) could be substituted into a partitioned form of eqs. (21). However, in some applications, the multi-point constraints can change many times during the course of analysis. This can be due to lock-up of certain members in a deployment analysis, contact conditions as in assembly, docking and berthing or articulation of structural components. In such cases it may be more computationally efficient to eliminate the j th dependent degree-of-freedom by multiplying the j th row of [M], (G), (f) and (F) by c_{jk}, (where c_{jk} is an entry of the constraint matrix), and adding the result to all the other rows of [M], (G), (f) and (F) respectively, for all k and j in the ranges $1 \leq k \leq N_D$ and $1 \leq j \leq N_C$. This effectively eliminates (δd_D). A similar procedure is followed for the columns of [M] thereby effectively eliminating (d_D).

Constraints Due to Rigid Members

Axial rigidity

The i^{th} member is defined as axially rigid if the neutral axis strain, its variation and its derivatives vanish. Then from eqs (11),

$$\delta\eta_{i,1} = \delta\eta_{i,2} + \delta\Delta\xi_i\tan\theta_i \tag{23a}$$

$$\dot{\eta}_{i,1} = \dot{\eta}_{i,2} + \Delta\dot{\xi}_i\tan\theta_i \tag{23b}$$

where, for simplicity, the first-order nonlinear term of eq. (11) caused by local finite element flexure has been dropped. Differentiation with respect to time of eq. (23b) yields,

$$\ddot{\eta}_{i,1} = \ddot{\eta}_{i,2} + \Delta\ddot{\xi}_i\tan\theta_i + \Delta\dot{\xi}_i\dot{\theta}_i\sec^2\theta_i \tag{23c}$$

and from eqs. (7) and (23b)

$$\dot{\theta}_i = \Delta\dot{\xi}_i/(L_i\cos\theta_i)$$

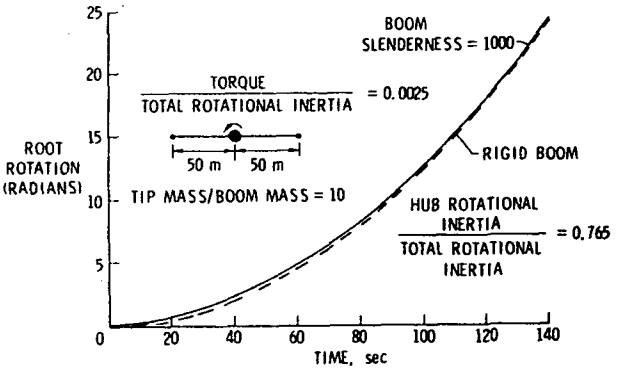


Figure 3. Hub Rotation Due To Applied Torque Step Load

Equations (23a) and (23c) may be used to eliminate $\delta\eta_{j,1}$ and $\eta_{j,1}$ in eq. (21) by appropriate manipulation of the rows of (G), (f) and (F), and the rows and columns of [M]. The nonlinear term in eq. (23c) cannot be dropped as it is not a negligible term. It may be handled through appropriate augmentation of the internal force vector (f).

Flexural rigidity

If the i^{th} member is flexurally rigid then, from eq. (18),

$$\delta\phi_{i,1} = \delta\phi_{i,2}$$

$$\dot{\phi}_{i,1} = \dot{\phi}_{i,2}$$

and from the second of eqs. (7),

$$\delta\xi_{i,1} = \delta\xi_{i,2} - \delta\Delta\eta_i\tan\theta_i - L_i\delta\phi_{i,2}/\cos\theta_i \tag{24a}$$

$$\dot{\xi}_{i,1} = \dot{\xi}_{i,2} - \Delta\dot{\eta}_i\tan\theta_i - L_i\dot{\phi}_{i,2}/\cos\theta_i \tag{24b}$$

Differentiation of eq. (24b) with respect to time yields,

$$\ddot{\xi}_{i,1} = \ddot{\xi}_{i,2} - \Delta\ddot{\eta}_i\tan\theta_i - L_i\ddot{\phi}_{i,2}/\cos\theta_i - \dot{\varepsilon}_{i,m}\theta_i L_i\cos\theta_i \tag{24c}$$

Eqs. (24a) and (24c) may be used to eliminate $\delta\phi_{i,1}$, $\dot{\phi}_{i,1}$, $\delta\xi_{i,1}$, $\ddot{\xi}_{i,1}$ in eq. (21) by appropriate manipulation of the rows of (G), (f), and (F) and the rows and columns of [M]. The nonlinear term in eqs. (24c) cannot be dropped as it is not negligible. It may be handled through appropriate augmentation of the internal force vector (f).

Numerical Solution of Equations of Motion

Equation (24) may be solved numerically by selecting from any of a wide choice of time integration algorithms. (See for example the survey article of reference 6.) An explicit algorithm is chosen here because it can readily accommodate a host of nonlinearities, damping mechanisms and control actuators. In particular a modified central difference algorithm is selected. This leads to the following recursive procedure,

$$(\ddot{d}(t)) = -[[M] + (1/2)\Delta t[D]]^{-1}\{(f(d(t))) + (G(d(t),\dot{d}_p(t),\ddot{d}_p(t))+[D](\dot{d}(t-\Delta t))-(F)\}$$

$$(\dot{d}(t)) = (\dot{d}(t-\Delta t)) + (1/2)\Delta t(\ddot{d}(t)+\ddot{d}(t-\Delta t))$$

$$(d(t)) = (d(t-\Delta t)) + \Delta t(\dot{d}(t)) \tag{25}$$

where (d_p) and $(\dot{d}_p)$ are velocity and acceleration predictors which are used only in conjunction with terms on the order of the flexure of a finite element. The predictors are chosen herein as,

$$\dot{d}_p(t) = \dot{d}(t-\Delta t) + \Delta t\ddot{d}(t-\Delta t) \tag{26}$$

$$\ddot{d}_p(t) = \ddot{d}(t-\Delta t) \tag{27}$$

and [D] is a linear viscous damping matrix.

Applications

The convected transient analysis presented herein is applicable to multi-body flexible truss and frame problems which may involve maneuver through large angles, translation through space or deployment. In this section the following problems are considered: slewing maneuver of a long slender boom, translation of two slender beams hinged together, deployment of an axisymmetric multi-flexible-member hoop and deployment of a planar multi-flexible member unfolding structure. In each application cubic flexural shape functions are chosen.

Slewing Maneuver

Figure 3 displays the time varying response of a slender boom with a rigid central hub subject to an applied torque. The hub rotation for a connected flexible boom with slenderness ratio 1000 is compared with that of a connected rigid boom. The hub rotation with a connected flexible boom is always equal to or greater than that of a connected rigid boom. This occurs because the inertial mass accelerated by the applied torque is the sum of the hub inertial mass plus the inertial mass of the boom from the hub out to the flexural wave front and the latter mass is less than or equal to the total boom inertial mass. Hence, the variation of the hub motion with connected rigid and flexible booms depends on the relative rotational inertial masses of the hub and boom as well as boom slenderness ratio. For the case displayed in figure 3, the inertial mass of the hub is considerably greater than that of the boom. Hence the deviation between the rigid and flexible responses is small.

As shown in figure 4, the tip motions of the flexible and rigid booms are quite different from that of the hub. Specifically, the tip motion with the flexible boom lags that with the rigid boom. This may be understood by considering the time delay caused by flexural wave propagation along the boom. As might be expected, the difference between rigid and flexible boom tip motion is relatively large compared to the differences at the hub. Thus boom flexiblity can greatly influence pointing without significantly influencing hub rotation. Finally, in figure 5, the axial load experienced by the flexible and rigid booms is compared. Axial loads are due to centrifugal

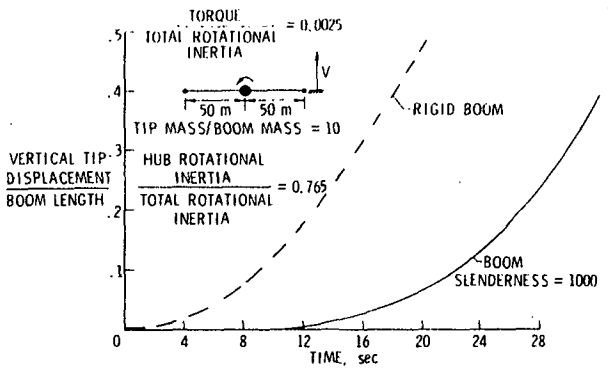


Figure 4. Boom Tip Displacement Due To Applied Torque Step Load

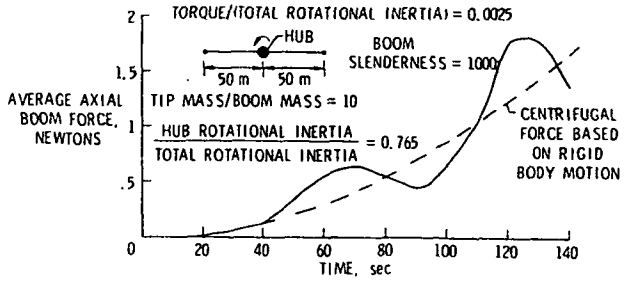


Figure 5. Axial Load Due To Applied Torque Step Load

force. The flexible boom axial load is seen to oscillate about the rigid value with increasing amplitude and frequency as the boom gains angular velocity.

Translation of two hinged flexible beams by a constant-direction step load

Figures 6a and 6b display the calculated motion of two identical beams hinged together at their ends and subject to constant-direction step load at one end. The problem is kinematically nonlinear because the torque caused by the constant-directional load and the load components in the beam modes vary with the kinematics. Displacements are shown in figure 6a for a flexible beam and the trajectory of the load application point is shown in figure 6b for both a flexible and a rigid beam. Examination of the trajectories of the load application point indicate that flexure has little effect on trajectory during the early phase of motion, but becomes more important as the motion proceeds.

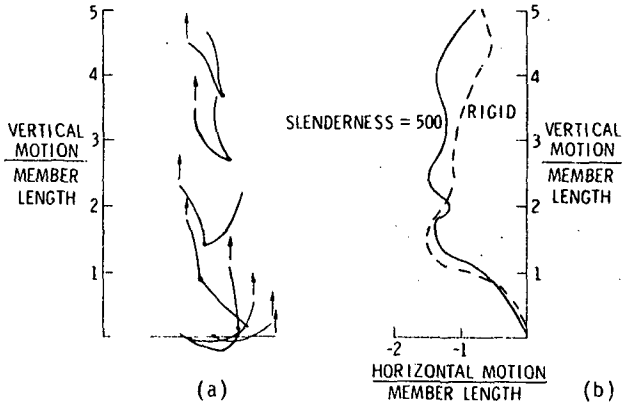


Figure 6. Large Distortion and Motion of Two Pin-Connected Beams Subject to a Vertical Tip Step Load
(a) Member Deformations
(b) Load Point Trajectory

Hoop Deployment

The hoop investigated here is similar in some respects to that of the Hoop/Column Antenna concept of reference 7. In the present study the complex control linkage between adjacent members of the hoop in reference 7 is replaced by

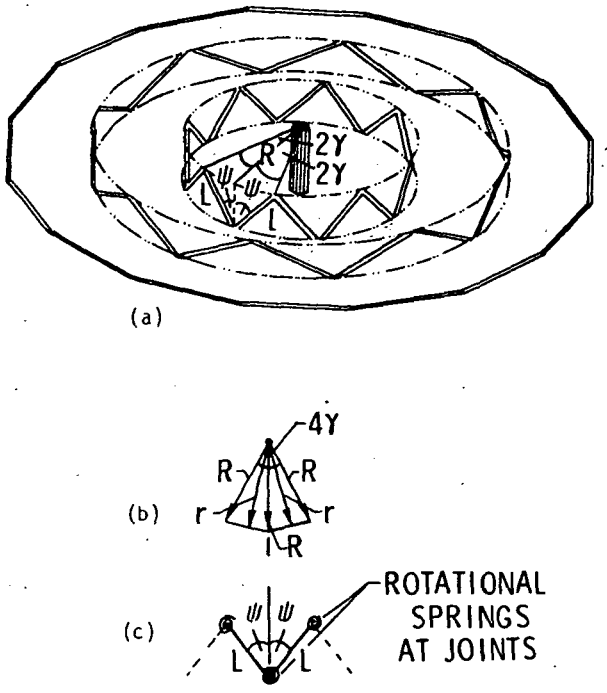


Figure 7. Hoop Deployment (N=16)
(a) Deployment Sequence
(b) Top View of Two Adjacent Members
(c) Front View of Two Adjacent Members

simple linear rotational springs at each joint which drive the deployment as shown in figures 7a and b. The hoop is assumed to be composed of N, (an even integer), identical flexible beam members of length L. The members are assumed to be connected through universal joints which do not transmit any moments. At any time during deployment, the members project on the horizontal plane as an N sided polygon with the joints between adjacent members lying on a circle of radius R. As deployment proceeds, R increases to a fully deployed hoop values of R_h.

Due to the symmetry of the hoop, only one member of the hoop, say the i^{th} member, needs to be analyzed provided the effects of the other members on it are accounted for. The effect of the other members on the i^{th} member can be systematically derived by establishing the virtual work of all the members in terms of the motions of the i^{th} member. Effectively, the other members constrain the motion of the i^{th} member and it is shown herein that this constraint may be modeled as an additional mass matrix superimposed on the mass matrix of the i^{th} member.

As shown in figure 7b, the i^{th} member lies in a plane normal to and translating parallel to the local radius r. Thus its virtual work has a contribution from the motion of the member in the plane and a contribution from the motion of the plane, namely,

$$\delta W_i = \delta W_i^{(i)} + \delta W_i^{(2)}$$

The first term is associated with member motion in the plane and has already been established in eq. (17). It remains to establish the second term. The second term is derived from the geometric relationship between the translations of the plane and motions of the i^{th} member in the plane. For the purpose of deriving this geometric relationship, the members are assumed to behave rigidly. However, this assumption is relaxed later in performing the dynamic deployment analysis.

The angle 2γ which is formed by the radii of two adjacent joints depends on the number of hoop members as

$$\gamma = \pi/N$$

The value of α remains constant during deployment, but as R increases during the deployment process, the angle ψ formed by the member and the vertical axis varies since,

$$R\sin\gamma = (L/2)\sin\psi \qquad (27)$$

When fully deployed ψ is $\pi/2$ and

$$R_h = L/(2\sin\gamma) \qquad (28)$$

Since,

$$\sin\psi = (n_b - n_a)/L$$

where n_a and n_b are horizontal displacements of the hoop member ends in a plane normal to r, eq. (27) yields,

$$R\sin\gamma = (1/2)(n_b - n_a) \qquad (29)$$

The contribution to the virtual work due to radial motion is,

$$\delta W_i^{(2)} = m\ddot{r}\delta r$$

Noting that

$$r = R\cos\gamma$$

and substituting from eq. (29) gives

$$\delta W_i^{(2)} = (m/4)(\ddot{n}_b - \ddot{n}_a)(\delta n_b - \delta n_a)\cot^2\gamma$$

The principle of virtual work applied to the entire hoop may be expressed as,

$$N\delta W_i^{(1)} + N(\delta d)^T[M_h](d) + 2N(\delta d)^T[K_s](d) = 0 \qquad (30)$$

where,

$$(d) = (n_{i,1}, \xi_{i,1}, \phi_{i,1}, n_{1,2}, \xi_{i,2}, \phi_{i,2}, \cdots$$
$$, n_{p,2}, \xi_{p,2}, \phi_{p,2})^T$$

and p is the number of finite elements employed to model the i_{th} hoop member. The nonzero entires of $[M_h]$ are,

$$(m_h)_{1,1} = (m_h)_{q,q} = (m/4)\cot^2\gamma$$

$$(m_h)_{1,q} = (m_h)_{q,1} = -(m/4)\cot^2\gamma$$

The nonzero entries of $[K_s]$ are

$$(k_s)_{1,3} = (k_s)_{6p,6p} = k$$

where k is the rotational spring constant at each joint. Dividing eq. (30) by N reveals that the equation of motion for axisymmetric deployment is identical to that for a single hoop member whose mass matrix is incremented by $[M_h]$ and which has grounded rotational springs at each end with a spring constant equal to twice the nominal spring constant value.

As an example application, consider a twenty member hoop (N=20). Figure 8 displays the predicted motion of two adjacent flexible hoop members during the deployment process. Each member has a slenderness ratio of 1000. Hoop member deformation is shown at one quarter, one-half and near fully deployed. During deployment, each member deforms into a two half-wave pattern which is consistent with the symmetry of the hoop. In figure 9, the time for the hoop to reach full deployment is examined over a range of N values. Two cases are displayed. In one case the individual hoop member length is fixed so that as N increases, the fully deployed hoop radius and total hoop weight also increase, while in the other case, the fully deployed hoop radius is fixed so that as N increases, the individual member lengths and mass decrease, however the total hoop weight remains essentially unchanged.

In each case results are shown for hoops containing all rigid or all flexible members and flexibility is seen to have little affect on deployment time.

The trends for the two cases are different and, where they cross, the hoop geometries are identical. The trend for each case may be approximately established in closed form by considering only the influence of the diagonal entries of the added mass matrix $[M_h]$ on a rigid member. Such a consideration reveals that for the fixed length case the deployment time varies linearly with N while for the fixed radius case it varies inversely with the square root of N. The deployment analysis confirms these trends.

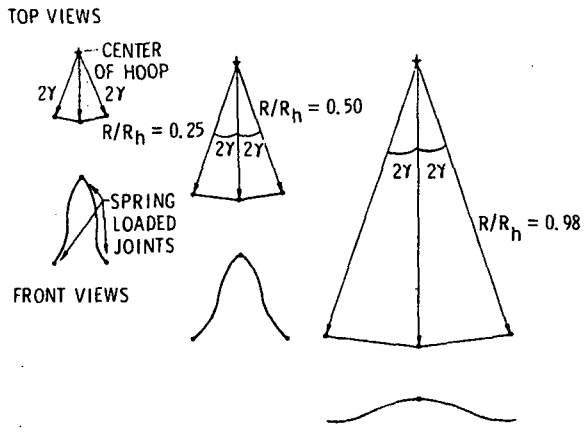


Figure 8. Top and Front Views of Two Adjacent Hoop Members During Deployment of Twenty Member Hoop, k=2.82 m.N/radian

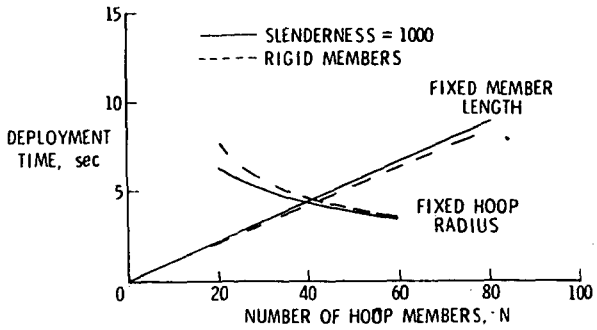


Figure 9. Deployment Times For Hoops Composed of Various Numbers of Members

Multi-member Unfolding Deployment

An accordian-like flexible beam deployable structure having N pin-connected members is shown in figure 10. When fully deployed the structure is a continuous flexible boom. Thus, when the angle between two adjacent members reaches 180 degrees, the joint is assumed locked and thereafter is a rigid joint. The deployment of the structure is driven by rotational springs located at each joint. All springs are identical, initially compressed and fully relaxed when the joint they are located at locks-up. Considerable insight into unfolding deployable structures is gained by examination of this generic structure.

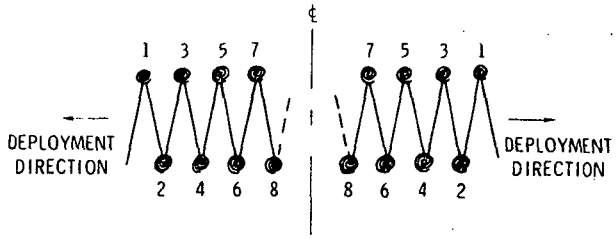


Figure 10. Geometric Configuration of Planar Unfolding Structure Showing Rotational Joint Springs and Joint Numbering Pattern. (Not An Actual Deployment)

Even Number of Members (N=2,4,6,...)

N=2 - Figure 11 displays the calculated deployment sequence of a two-member structure in a cumulative superimposed fashion where each member was modeled with two finite elements. Member deformations have not been scaled-up, but are consistent with the drawn member length and little flexure of the members prior to lock-up is visible. Following lock-up considerable member flexure occurs and the deployed boom vibrates in essentially a first symmetric free-free mode shape.

N=4 - Figure 12 displays the deployment of a four member structure in the same fashion as that used in figure 11. Prior to lock-up, members appear to exhibit more flexure than in the single-member case. Since the outer members accelerate less mass, the outer joints lock-up first shortly followed by the inner joint. Following lock-up,

the deployed boom vibrates with essentially a second symmetric free-free mode shape. However, the post-lock-up shape depends upon member stiffness. Though not shown, stiffer members can yield a first symmetric free-free vibration shape.

N=6,8,10,...When the structure has six or more members, the time difference between the lock-up of the joints becomes greater than that exhibited in the two-member case. Lock-up begins at the outer most joints and proceeds towards the center of the structure as shown, for example, in the progressive deployment states of the sixteen-member structure of figure 13. Also, lock-up times for some of the joints in various multi-member configurations are provided in the Table.
Joints are numbered beginning at the outermost joint and progressing towards the center so that, independent of the number of members, the outermost joint is always joint number one. Note that the lock-up time for joint 1 is independent of the number of members and that other joint lock-up times also become independent of the number of members as the number of members increases. In general, lock-up times for all but the innermost joints are independent of the number of members. The result is that the structural joints lock-up sequentially even though all members and springs are identical. Again the reason for this is traced to the quantity of mass the various members must accelerate.

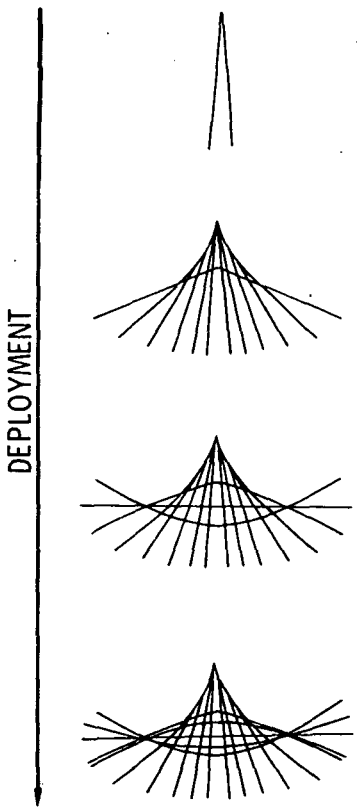


Figure 11. Deployment Sequence of Two-Flexible-Member Unfolding Structure

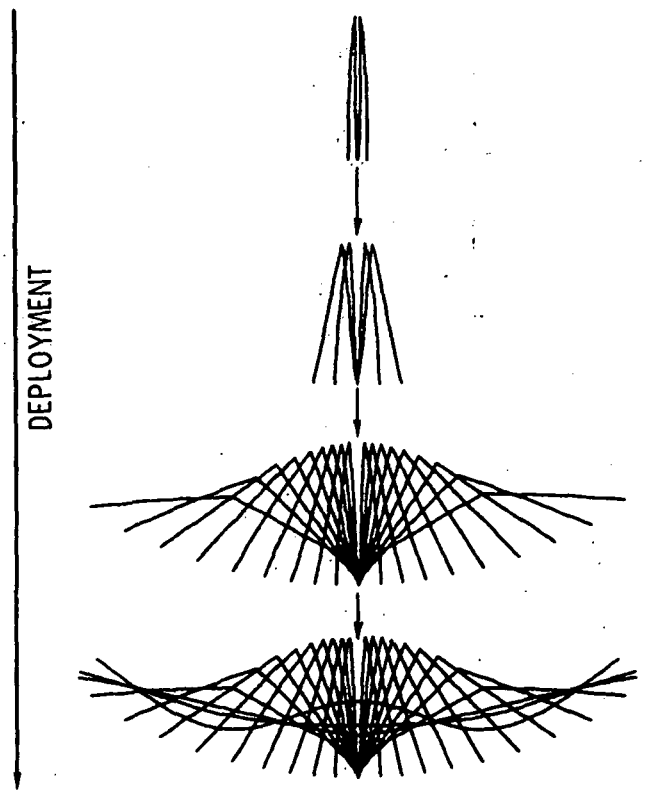


Figure 12. Deployment Sequence of Four-Flexible-Member Unfolding Structure

This basically sequential pattern is considered a desirable quality for deployable space structures. (See references 8 and 9.) For the multi-member structure considered here, sequential deployment appears to occur naturally; that is, without the aid of external mechanisms, controls or dampers. It is anticipated that multi-member deployment configurations other than the one considered here will also exhibit natural deployment patterns, though they may not be of the sequential type. If this is the case, it would allow the designer to select member and spring properties to achieve a preferred deployment pattern without the expensive addition of controlled or passive mechanisms to enforce a prescribed pattern.

Odd Number of Members (N=3,5,7,...)

For the unfolding type structures considered here the deployment pattern for an odd number of members differs from that of an even number of members. For example, consider the five member structure of figure 14. Due to the odd number of members, the deployment pattern tends to appear rotated counter-clockwise. Actually, the two outer members are rotating clockwise while the three inner members are rotating counter-clockwise. The result is that the pattern satisfies the conservation of angular momentum. As in the case of an even number of members, the deployment pattern tends to become sequential as the number of members in the structure increases. However, the pattern will be distorted somewhat due to the conservation of angular momentum. This distortion should decrease as the number of members increases.

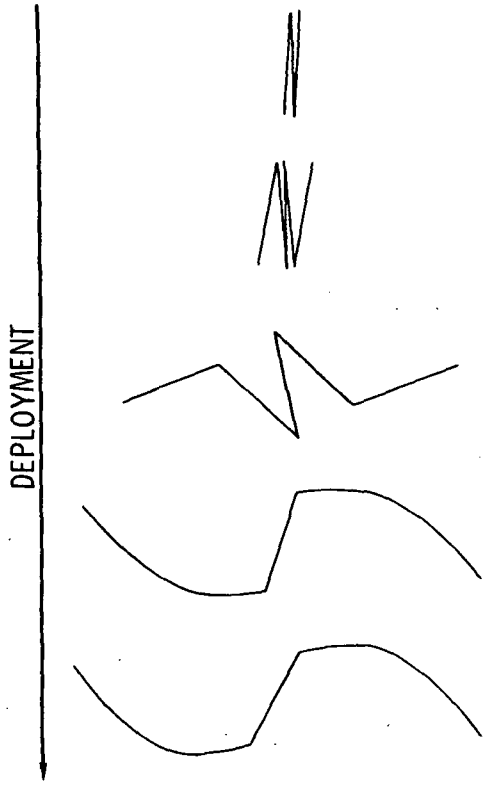


Figure 13. Deployment Sequence of Sixteen-Flexible-Member Unfolding Structure

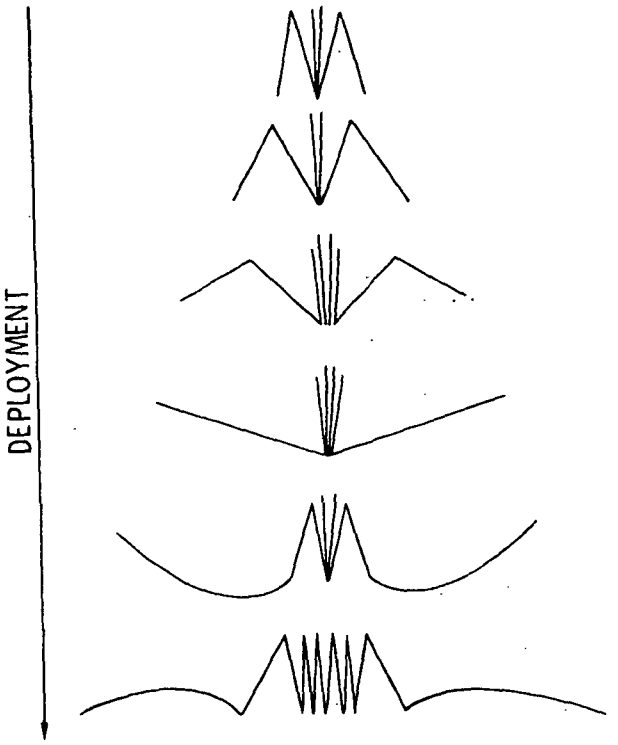


Figure 14. Deployment Sequence of Five-Flexible-Member Unfolding Structure

Concluding Remarks

The application of convected finite element transient analysis to maneuver and deployment of two-dimensional truss and frame space structures has been presented. This approach permits the utilization of a vast storehouse of finite element computational technology and, within the finite element framework, the convected analysis accounts for both large rotations and large elastic deformations.

A consistent mass (rather than a lumped mass) formulation of the equations of motion has been incorporated so that exact rotational inertial properties are retained in the analysis which is critical for space structures which can undergo large rotations. Within the convected analysis framework, a consistent mass approach has been shown to allow the resulting inertial terms to be ordered in relative magnitude and many of the terms which require frequent updating may often be safely neglected when the finite element grid is a relatively fine one. For the problems considered herein, little or no updating was required even for booms with slenderness ratios of 1000 when two finite elements were used per halfwave of deformation. Thus, frequent up-dating, a major shortcoming of the consistent mass approach in a convected coordinate system can be avoided.

Results of the convected analysis have been presented for maneuvering of slender booms through large rotations, translation of two hinged beams, deployment of an unfolding multi-flexible-member axisymmetric hoop and deployment of a planar unfolding multi-flexible-member structure. For a symmetrical arrangement of booms extended from a central hub and subject to a torque, flexibility influences boom tip deflection and hence pointing, but has little influence on hub rotation.

Deployment of an axisymmetric multi-flexible-member hoop may be analyzed by considering only one hoop member when an appropriate derived mass matrix is superimposed on the member mass matrix. Applying this procedure some aspects of hoop deployment have been studied. It has been found that hoop deployment time is essentially independent of member flexibility and that flexible members deform in a two-half-wave pattern.

Furthermore, planar unfolding deployment of a multi-flexible-member structure has indicated that these structures may possess natural deployment patterns which are very desirable in practice. The structure considered herein displays a natural sequential deployment pattern without the use of any control mechanisms which are usually quite complicated and expensive.

Number of Members	Joint Number				
	1	2	3	4	5
4	1.448	1.468			
8	1.453	2.308	2.792	2.814	
12	1.452	2.330	2.856	3.631	3.930
16	1.454	2.401	2.861	3.680	4.643
20	1.461	2.476	2.877	3.816	-

Table. Joint Lock-Up Times In Seconds For An Even Number Of Deploying Members

REFERENCES

1. Belytschko, T. and Hsieh, B.J.: Nonlinear Transient Finite Element Analysis With Convected Coordinates. International Journal for Numerical Methods in Engineering. Vol. 7, 1973, pp. 255-272.

2. Belytschko, T.; Welch, R.E.; and Bruce, R.W.: Large Displacement Nonlinear Transient Analysis by Finite Elements. Proceedings of International Conference on Vehicle Structural Mechanics. SAE, Warrendale Pa., 1974, pp. 188-197.

3. Belytschko, T. and Schwew, L.: Large Displacement Transient Analysis of Space Frames. International Journal for Numerical Methods in Engineering. Vol 11, 1977, pp. 65-84.

4. Housner, J.M. and Knight, N. F., Jr.: On the Dynamic Collapse of a Column Impacting a Rigid Surface. AIAA Journal, Aug. 1983, Vol. 20, pp. 1187-1195.

5. Laurenson, R.M.: Influence of Mass Representation on the Modal Analysis of Rotating Flexible Structures. Proceedings of the 24 th AIAA SDM, Lake Tahoe NV, May 1983, Paper No. 83-0915.

6. Park, K.C.: Time Integration of Structural Dynamics Equations: A Survey. ASME Preesure Vessels and Piping: Design Technology. 1982 edition, pp.277-291.

7. Sullivan, M. R.: LSST (Hoop/Column) Maypole Antenna Development Program. NASA CR 3558, Parts 1 and 2, June 1982.

8. Hedgepeth, J.M.:Sequential Deployment of Truss Structures. Large Space Systems Technology - 1981, NASA CP 2215, Part 1, pp 179-192, Nov. 1981.

9. Schwartzberg, F.R.; Coyner, J.V., Jr.; and Tobey, W.H.: Development and Application of Space-Deployable Box Truss Structures. Presented at 32nd Congress of the International Astronautical Federation, U. of Rome, Rome Italy, Sept. 6-12, 1981.

Appendix A

This appendix contains entries of the vector $(\hat{f}_i)$.

$$f_{i,1} = -P_i c - (T_{i,1} + T_{i,2})s/L_i + P_i(\gamma_{i,1} + \gamma_{i,2})s$$

$$f_{i,2} = -P_i s + (T_{i,1} + T_{i,2})c/L_i - P_i(\gamma_{i,1} + \gamma_{i,2})c$$

$$f_{i,3} = T_{i,1} + P_i\gamma_{i,1}$$

$$f_{i,4} = -f_{i,1}$$

$$f_{i,5} = -f_{i,2}$$

$$f_{i,6} = T_{i,2} + P_i\gamma_{i,2}$$

where,

$$T_{i,1} = -(EI)_i h''(0)$$

$$T_{i,2} = (EI)_i h''(1)$$

$$\gamma_{i,1} = (\Gamma_{i,11} + \Gamma_{i,21})\phi_{i,1} + (\Gamma_{i,12} + \Gamma_{i,22})\phi_{i,2} - (\Gamma_{i,11} + \Gamma_{i,12} + \Gamma_{i,21} + \Gamma_{i,22})(\theta_i - \alpha_i)$$

$$\Gamma_{i,11} = \int_0^1 a_{i,1}a''_{i,1}d\bar{s}; \quad \Gamma_{i,12} = \int_0^1 a_{i,1}a''_{i,2}d\bar{s}$$

$$\Gamma_{i,21} = \int_0^1 a_{i,2}a''_{i,1}d\bar{s}; \quad \Gamma_{i,22} = \int_0^1 a_{i,2}a''_{i,2}d\bar{s}$$

$$c = \cos\theta_i$$

$$s = \sin\theta_i$$

Appendix B

This appendix contains entire of the matrices $[\hat{M}_i^{(j)}]$; j=1,2,3 and the vectors $(\hat{G}_i^{(k)})$; k=1,2,3,4 .

$$m_{11}^{(1)} = m_{22}^{(1)} = m_{44}^{(1)} = m_{55}^{(1)} = 1/3$$

$$m_{14}^{(1)} = m_{25}^{(1)} = 1/6$$

$$m_{i,11}^{(2)} = [2(e_{i,1} - e_{i,s}) + e_{i,2}]s^2$$

$$m_{i,12}^{(2)} = -[2(e_{i,1} - e_{i,s}) + e_{i,2}]sc$$

$$m_{i,13}^{(2)} = (A_{i,1s} - A_{i,1} - A_{i,12} - A_{i,11})s$$

$$m_{i,14}^{(2)} = (2e_{i,s} - e_{i,1} - e_{i,2})s^2$$

$$m_{i,15}^{(2)} = -(2e_{i,s} - e_{i,1} - e_{i,2})sc$$

$$m_{i,16}^{(2)} = (A_{i,2s} - A_{i,2} - A_{i,12} - A_{i,22})s$$

$$m_{i,22}^{(2)} = (2e_{i,1} - 2e_{i,s} + e_{i,2})c^2$$

$$m_{i,23}^{(2)} = (A_{i,1} - A_{i,1s} + A_{i,11} + A_{i,12})c$$

$$m_{i,24}^{(2)} = (e_{i,1} + 2e_{i,2} - 2e_{i,s})sc$$

$$m_{i,25}^{(2)} = (2e_{i,s} - e_{i,1} - e_{i,2})c^2$$

$$m_{i,26}^{(2)} = (A_{i,2} - A_{i,2s} + A_{i,12} + A_{i,22})c$$

$$m_{i,33}^{(2)} = A_{i,11}$$

$$m_{i,34}^{(2)} = (A_{i,11} + A_{i,12} - A_{i,1s})s$$

$$m_{i,35}^{(2)} = (A_{i,1s} - A_{i,11} - A_{i,12})c$$

$$m_{i,36}^{(2)} = A_{i,12}$$

$$m_{i,44}^{(2)} = (e_{i,2} - 2e_{i,s})s^2$$

$$m^{(2)}_{i,45} = -(e_{i,2} - 2e_{i,s})cs$$

$$m^{(2)}_{i,46} = (A_{i,12} + A_{i,22} - A_{i,2s})s$$

$$m^{(2)}_{i,55} = (e_{i,2} - 2e_{i,s})c^2$$

$$m^{(2)}_{i,56} = (A_{i,2s} - A_{i,12} - A_{i,22})c$$

$$m^{(2)}_{i,66} = A_{i,22}$$

$$m^{(3)}_{i,11} = 2(H_{i,s} - H_{i,1})sc + H_{i,2}s^2$$

$$m^{(3)}_{i,12} = (H_{i,1} - H_{i,s})(c^2 - s^2) - H_{i,2}sc$$

$$m^{(3)}_{i,14} = (H_{i,1} - 2H_{i,s})sc - H_{i,2}s^2$$

$$m^{(3)}_{i,15} = H_{i,s}(c^2 - s^2) - H_{i,1}c^2 + H_{i,2}sc$$

$$m^{(3)}_{i,22} = 2(H_{i,1} - H_{i,s})sc - H_{i,2}sc$$

$$m^{(3)}_{i,24} = H_{i,1}s^2 + H_{i,s}(c^2 - s^2) + H_{i,2}sc$$

$$m^{(3)}_{i,25} = 2H_{i,s}sc - H_{i,2}c^2$$

$$m^{(3)}_{i,44} = 2H_{i,s}sc + H_{i,3}s^2$$

$$m^{(3)}_{i,45} = -H_{i,s}(c^2 - s^2) - H_{i,3}sc$$

$$m^{(3)}_{i,55} = -2H_{i,s}sc$$

where

$$e_{i,1} = A_{i,1} + A_{i,2}$$

$$e_{i,2} = A_{i,11} + 2A_{i,12} + A_{i,22}$$

$$e_{i,s} = A_{i,1s} + A_{i,2s}$$

$$A_{i,1} = \frac{1}{L_i}\int_0^1 a_{i,1}d\bar{s}; \quad A_{i,2} = \frac{1}{L_i}\int_0^1 a_{i,2}d\bar{s}$$

$$A_{i,11} = \frac{1}{L_i^2}\int_0^1 a_{i,1}^2 d\bar{s}; \quad A_{i,12} = \frac{1}{L_i^2}\int_0^1 a_{i,1}a_{i,2}d\bar{s};$$

$$A_{i,11} = \frac{1}{L_i^2}\int_0^1 a_{i,2}^2 d\bar{s}$$

$$A_{i,1s} = \frac{1}{L_i}\int_0^1 \bar{s}a_{i,1}d\bar{s}; \quad A_{i,2s} = \frac{1}{L_i}\int_0^1 \bar{s}a_{i,2}d\bar{s}$$

$$H_{i,1} = A_{i,1}\phi_{i,1} + A_{i,2}\phi_{i,2} - (A_{i,1} + A_{i,2})(\theta_i - \alpha_1)$$

$$H_{i,s} = A_{i,1s}\phi_{i,1} + A_{i,2s}\phi_{i,2} - (A_{i,1s} + A_{i,2s})(\theta_i - \alpha_i)$$

$$H_{i,2} = A_{i,11}\phi_{i,1}^2 + A_{i,22}\phi_{i,2}^2 + (A_{i,11} + 2A_{i,12} + A_{i,22})(\theta_i - \alpha_i)^2 + 2A_{i,12}\phi_{i,1}\phi_{i,2} - 2(A_{i,11}+A_{i,12})(\theta_i-\alpha_i)\phi_{i,1} - 2(A_{i,12} + A_{i,22})\phi_{i,2}(\theta_i - \alpha_i)$$

$$g^{(1)}_{i,1} = (e_{i,1} + e_{i,2} - e_{i,s})s\dot{\theta}_i\dot{\varepsilon}_{m,i}$$

$$g^{(1)}_{i,2} = -(e_{i,1} + e_{i,2} - e_{i,s})c\dot{\theta}_i\dot{\varepsilon}_{m,i}$$

$$g^{(1)}_{i,3} = -(A_{i,11} + A_{i,12})\dot{\theta}_i\dot{\varepsilon}_{m,i}$$

$$g^{(1)}_{i,4} = (e_{i,s} - e_{i,2})s\dot{\theta}_i\dot{\varepsilon}_{m,i}$$

$$g^{(1)}_{i,5} = -(e_{i,s} - e_{i,2})c\dot{\theta}_i\dot{\varepsilon}_{m,i}$$

$$g^{(1)}_{i,6} = -(A_{i,12} + A_{i,22})\dot{\theta}_i\dot{\varepsilon}_{m,i}$$

$$g^{(2)}_{i,1} = [2c\dot{H}_{i,3} + H_{i,3}(s\dot{\theta}_i - c\dot{\varepsilon}_{m,i}) + sH_{i,2}\dot{\varepsilon}_{m,i}]\dot{\theta}_i$$

$$g^{(2)}_{i,2} = -[2s\dot{H}_{i,3} + H_{i,3}(c\dot{\theta}_i + s\dot{\varepsilon}_{m,i}) + cH_{i,2}\dot{\varepsilon}_{m,i}]\dot{\theta}_i$$

$$g^{(2)}_{i,3} = g^{(2)}_{i,6} = 0$$

$$g^{(2)}_{i,4} = [-2c\dot{H}_{i,s} + H_{i,s}(s\dot{\theta}_i - c\dot{\varepsilon}_{m,i}) - sH_{i,2}\dot{\varepsilon}_{m,i}]\dot{\theta}_i$$

$$g^{(2)}_{i,5} = [c\dot{H}_{i,2}\dot{\varepsilon}_{m,i} - 2s\dot{H}_{i,s} + H_{i,s}(c\dot{\theta}_i + s\dot{\varepsilon}_{m,i})]\dot{\theta}_i$$

where

$$H_{i,3} = H_{i,1} - H_{i,2}$$

$$c = \cos\theta_i; \quad s = \sin\theta_i$$

$$g^{(3)}_{i,1} = s(H_{i,s}\dot{\varepsilon}^*_{m,i} - H_{i,1}\ddot{\bar{\eta}}^{**}_{i,1}) + (sH_{i,2} - cH_{i,3})\dot{\theta}^*_i + [H_{i,4}\phi_{i,1} + H_{i,5}\phi_{i,2} - (H_{i,4} + H_{i,5})(\theta_i - \alpha_i)]s\dot{\theta}^*_i$$

$$g^{(3)}_{i,2} = c(H_{i,s}\dot{\varepsilon}^*_{m,i} + H_{i,1}\ddot{\bar{\eta}}^{*}_{i,1}) - (sH_{i,3} + cH_{i,2})\dot{\theta}^*_i - [H_{i,4}\phi_{i,1} + H_{i,5}\phi_{i,2} - (H_{i,4} + H_{i,5})(\theta_i - \alpha_i)]c\dot{\theta}^*_i$$

$$g^{(3)}_{i,3} = g^{(3)}_{i,6} = 0$$

$$g^{(3)}_{i,4} = s(H_{i,s}\dot{\varepsilon}^{*}_{m,i} + H_{i,1}\ddot{\bar{\eta}}^{**}_{i,1}) - (cH_{i,s} + sH_{i,2})\dot{\theta}^{*}_{i}$$

$$- [H_{i,4}\phi_{i,1} + H_{i,5}\phi_{i,2}$$

$$- (H_{i,4} + H_{i,5})(\theta_i - \alpha_i)]s\dot{\theta}^{*}_{i}$$

$$g^{(3)}_{i,5} = -c(H_{i,s}\dot{\varepsilon}^{*}_{m,i} + H_{i,1}\ddot{\bar{\eta}}^{**}_{i,1}) + (cH_{i,2} - sH_{i,s})\dot{\theta}^{*}_{i}$$

$$+ [H_{i,4}\phi_{i,1} + H_{i,5}\phi_{i,2}$$

$$- (H_{i,4} + H_{i,5})(\theta_i - \alpha_i)]c\dot{\theta}^{*}_{i}$$

where,

$$H_{i,4} = A_{i,11}\phi_{i,1} + A_{i,12}\phi_{i,2} - (A_{i,11}+A_{i,12})(\theta_i-\alpha_i)$$

$$H_{i,5} = A_{i,21}\phi_{i,1} + A_{i,22}\phi_{i,2} - (A_{i,21}+A_{i,22})(\theta_i-\alpha_i)$$

and where (*) denotes time differentiation with convected terms omitted, i.e.,

$$\dot{\varepsilon}^{*}_{m,i} = (c\Delta\ddot{\eta}_i + s\Delta\ddot{\xi}_i)/L_i$$

$$\dot{\theta}^{*}_{i} = (c\Delta\ddot{\xi}_i - s\Delta\ddot{\eta}_i)/L_i$$

$$\ddot{\bar{\eta}}^{**}_{i,1} = (c\ddot{\eta}_{i,1} + s\ddot{\xi}_{i,1})/L_i$$

$$g^{(4)}_{i,1} = s\dot{\theta}_i\{2(H_{i,4}\dot{\phi}_{i,1} + H_{i,5}\dot{\phi}_{i,2}) - (H_{i,4} + H_{i,5})\dot{\theta}_i$$

$$+ [H_{i,4}\phi_{i,1} + H_{i,5}\phi_{i,2}$$

$$- (H_{i,4} + H_{i,5})(\theta_i - \alpha_i)]\dot{\varepsilon}_{m,i}\}$$

$$g^{(4)}_{i,2} = -c\dot{\theta}_i\{2(H_{i,4}\dot{\phi}_{i,1} + H_{i,5}\dot{\phi}_{i,2}) - (H_{i,4} + H_{i,5})\dot{\theta}_i$$

$$+ [H_{i,4}\phi_{i,1} + H_{i,5}\phi_{i,2}$$

$$- (H_{i,4} + H_{i,5})(\theta_i - \alpha_i)]\dot{\varepsilon}_{m,i}\}$$

$$g^{(4)}_{i,3} = -H_{i,4}\dot{\theta}^2_i$$

$$g^{(4)}_{i,4} = -g^{(4)}_{i,1}$$

$$g^{(4)}_{i,5} = -g^{(4)}_{i,2}$$

$$g^{(4)}_{i,6} = -H_{i,5}\dot{\theta}^2_i$$

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-87635		

4. Title and Subtitle	5. Report Date
The LATDYN User's Manual	January 1986
	6. Performing Organization Code: 482-53-53-34
7. Author(s): Jerrold M. Housner, Paul E. McGowan, A. Louis Abrahamson and Michael G. Powell	8. Performing Organization Report No.
9. Performing Organization Name and Address: NASA Langley Research Center, Hampton, VA 23665-5225	10. Work Unit No.
	11. Contract or Grant No.
12. Sponsoring Agency Name and Address: National Aeronautics and Space Administration, Washington, DC 20546-0001	13. Type of Report and Period Covered: Technical Memorandum
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This manual presents the capabilities and instructions for the LATDYN (Large Angle Transient DYNamics) computer program. The LATDYN program is a tool for analyzing the controlled or uncontrolled dynamic transient behavior of interconnected deformable multi-body systems which can undergo large angular motions of each body relative other bodies. The program accommodates large structural deformation as well as large rigid body rotations and is applicable, but not limited to, the following areas:

(1) deployment of large flexible space structures
(2) slewing of large space structure components
(3) mechanisms with rigid or elastic components
(4) robotic manipulations of beam members

Presently the program is limited to two dimensional problems, but in many cases, three dimensional problems can be exactly or approximately reduced to two dimensions. The program uses convected finite elements to affect the large angular motions involved in the analysis. General geometry is permitted. Detailed user input and output specifications are provided and discussed with example runstreams. To date, LATDYN has been configured for CDC/NOS and DEC VAX/VMS machines. All coding is in ANSII-77 FORTRAN. Detailed instructions regarding interfaces with particular computer operating systems and file structures are provided.

17. Key Words (Suggested by Author(s))		18. Distribution Statement
Space structures	Multi-body dynamics	Unclassified-Unlimited
Deployment	Impact	
Robotics	Convected coordinates	
Mechanisms	Transient analysis	Subject Category 39
	Finite elements	
	Dynamic simulation	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	211	A10

For sale by the National Technical Information Service, Springfield, Virginia 22161